

625-CD-011-001

EOSDIS Core System Project

ECS Project Training Material Volume 11: Database Administration

December 1997

Hughes Information Technology Systems
Upper Marlboro, Maryland

ECS Project Training Material Volume 11: Database Administration

December 1997

Prepared Under Contract NAS5-60000
CDRL Item #129

RESPONSIBLE ENGINEER

Kenneth L. Prickett /s/ 12/17/97

Kenneth L. Prickett Date
EOSDIS Core System Project

SUBMITTED BY

Thomas A. Hickey /s/ 12/17/97

Tom Hickey, M&O Manager Date
EOSDIS Core System Project

Hughes Information Technology Systems
Upper Marlboro, Maryland

This page intentionally left blank.

Preface

This document is a contract deliverable with an approval code of 3. As such, it does not require formal Government approval. This document is delivered for information only, but is subject to approval as meeting contractual requirements.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Systems
1616 McCormick Dr.
Upper Marlboro, MD 20774-5372

This page intentionally left blank.

Abstract

This is Volume 11 of a series of lessons containing the training material for Version 2.0 Drop 2 of the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). This lesson provides a detailed description of the process required to perform the tasks associated with database administration.

Keywords: training, database administration, course objective, metadata

This page intentionally left blank.

Change Information Page

List of Effective Pages			
Page Number		Issue	
Title		Original	
iii through xii		Original	
1 through 74		Original	
Slide Presentation 1 through 39		Original	
Document History			
Document Number	Status/Issue	Publication Date	CCR Number
625-CD-011-001	Original	December 1997	

This page intentionally left blank.

Contents

Preface

Abstract

Introduction

Identification	1
Scope	1
Purpose.....	1
Status and Schedule	1
Organization	1

Related Documentation

Parent Document.....	3
Applicable Documents	3
Information Documents	3
Information Documents Referenced.....	3
Information Documents Not Referenced.....	3

Database Administration

Lesson Overview.....	7
Lesson Objectives.....	7
Importance.....	9
Special Instructions on Procedures.....	9

Overview

DBA Tasks and Functions	11
Sybase Directory Structure.....	11
Naming Conventions.....	12

Version 2.0 Databases

Version 2.0 Databases	15
-----------------------------	----

Maintaining the Interface File

Adding an entry to the interfaces file	17
Modifying an existing entry in the interfaces file	18

Starting and Stopping SQL Server

Start an SQL Server Process	21
Stop an SQL Server.....	21

Database Devices

Making Database Size Estimates and Planning	26
Preparing Database Unique Attributes (after a database has been created):	26

User Databases

User Databases.....	31
---------------------	----

Database Segments

Creating Segments.....	37
------------------------	----

Transaction Logs

What is a Transaction?.....	41
What is a Transaction Log?	41

How Transaction Logging Works	41
Getting Information about the Transaction Log	41

Backup and Recovery

Frequency and Schedule	43
Manual Database Backup Procedure	43
Database Recovery	44
Load Database and Load Transaction commands.....	46
BulkCopy Utility	46

SQL Server Login Accounts and Privileges

Receive Approval For Account Creation	49
Create SQL Server Login Accounts	51
Add User to Database(s)	51
Granting or Revoking Database Access Privileges.....	53

Database Tuning and Performance Monitoring

Configure SQL Server.....	55
Memory Utilization and Configuration.....	57

Database Security and Auditing

Database Security and Auditing.....	65
-------------------------------------	----

Integrity Monitoring

Integrity Monitoring	67
----------------------------	----

Troubleshooting

Diagnosing Database System Problems.....	69
Symptoms of a Damaged Master Database	69

Practical Exercises

Starting and Stopping SQL Server	71
Database Devices.....	71
User Databases.....	72
Backup and Recovery	72

Slide Presentation

Slide Presentation Description	73
--------------------------------------	----

Introduction

Identification

Training Material Volume 11 is part of Contract Data Requirements List (CDRL) Item 129, whose requirements are specified in Data Item Description (DID) 625/OP3 and is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract (NAS5-6000).

Scope

Training Material Volume 11 describes the processes and procedures required to accomplish Database Administration. This lesson is designed to provide the operations staff with sufficient knowledge and information to satisfy all lesson objectives.

Purpose

The purpose of this lesson is to provide a detailed course of instruction that forms the basis for understanding Database Administration. Lesson objectives are developed and will be used to guide the flow of instruction for this lesson. The lesson objectives will serve as the basis for verifying that all lesson topics are contained within this Student Guide and slide presentation material.

Status and Schedule

This lesson module provides detailed information about training for Version 2.0 Drop 2. Subsequent revisions will be submitted as needed.

Organization

This document is organized as follows:

Introduction:	The Introduction presents the document identification, scope, purpose, and organization.
Related Documentation:	Related Documentation identifies parent, applicable and information documents associated with this document.
Student Guide:	The Student Guide identifies the core elements of this lesson. All Lesson Objectives and associated topics is included.
Slide Presentation:	Slide Presentation is reserved for all slides used by the instructor during the presentation of this lesson.

This page intentionally left blank.

Related Documentation

Parent Document

The parent document is the document from which this ECS Training Material's scope and content are derived.

423-41-01	Goddard Space Flight Center, EOSDIS Core System (ECS) Statement of Work
-----------	---

Applicable Documents

The following documents are referenced within this ECS Training Material, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document:

420-05-03	Goddard Space Flight Center, Earth Observing System (EOS) Performance Assurance Requirements for the EOSDIS Core System (ECS)
423-41-02	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS)

Information Documents

Information Documents Referenced

The following documents are referenced herein and amplify or clarify the information presented in this document. These documents are not binding on the content of the ECS Training Material.

535-TIP-CPT-001	Goddard Space Flight Center, Mission Operations and Data Systems Directorate (MO&DSD) Technical Information Program Networks Technical Training Facility, Contractor-Provided Training Specification
609-CD-003-001	Operations Tools Manual
611-CD-004-001	Mission Operation Procedures for the ECS Project

Information Documents Not Referenced

The following documents, although not referenced herein and/or not directly applicable, do amplify or clarify the information presented in this document. These documents are not binding on the content of the ECS Training Material.

220-TP-001-001	Operations Scenarios - ECS Release B.0 Impacts
305-CD-020-002	Release B SDPS/CSMS System Design Specification Overview for the ECS Project

305-CD-021-002	Release B SDPS Client Subsystem Design Specification for the ECS Project
305-CD-022-002	Release B SDPS Interoperability Subsystem Design Specification for the ECS Project
305-CD-023-002	Release B SDPS Data Management Subsystem Design Specification for the ECS Project
305-CD-024-002	Release B SDPS Data Server Subsystem Design Specification for the ECS Project
305-CD-025-002	Release B SDPS Ingest Subsystem Design Specification [for the ECS Project
305-CD-026-002	Release B SDPS Planning Subsystem Design Specification for the ECS Project
305-CD-027-002	Release B SDPS Data Processing Subsystem Design Specification for the ECS Project
305-CD-028-002	Release B CSMS Communications Subsystem Design Specification for the ECS Project
305-CD-029-002	Release B CSMS System Management Subsystem Design Specification for the ECS Project
305-CD-030-002	Release B GSFC DAAC Design Specification for the ECS Project
305-CD-031-002	Release B Langley DAAC Design Specification for the ECS Project
305-CD-033-002	Release B EDC DAAC Design Specification for the ECS Project
305-CD-034-002	Release B ASF DAAC Design Specification for the ECS Project
305-CD-035-002	Release B NSIDC DAAC Design Specification for the ECS Project
305-CD-036-002	Release B JPL PO.DAAC Design Specification for the ECS Project
305-CD-037-002	Release B ORNL DAAC Design Specification for the ECS Project
305-CD-038-002	Release B System Monitoring and Coordination Center Design Specification for the ECS Project
305-CD-039-002	Release B Data Dictionary Subsystem Design Specification for the ECS Project
601-CD-001-004	Maintenance and Operations Management Plan for the ECS Project
604-CD-001-004	Operations Concept for the ECS Project: Part 1-- ECS Overview
604-CD-002-003	Operations Concept for the ECS Project: Part 2B -- ECS Release B
605-CD-002-001	Release B SDPS/CSMS Operations Scenarios for the ECS Project
607-CD-001-002	ECS Maintenance and Operations Position Descriptions

500-1002

Goddard Space Flight Center, Network and Mission Operations
Support (NMOS) Certification Program, 1/90

This page intentionally left blank.

Database Administration

Lesson Overview

This lesson will provide you with the tools needed to perform the various tasks required to administer and maintain the database and structure management for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS) during maintenance and operations.

Lesson Objectives

Overall Objective - This lesson provides a detailed description of the different tasks required to maintain the database and structure management for ECS, provide the operations interface to perform database administration utilities such as product installation and disk storage management, managing user accounts and privileges, backup and recovery, monitoring physical allocation of database resource information, loading metadata and maintaining metadata.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11 Database Administration* and a functioning system.

Standard - The student will perform without error the procedures required to perform database administration.

Specific Objective 1 - The student will be able to create new database devices, allocate appropriate disk space to house the new database, and maintain database segments including:

- Indexing physical allocation of databases.
- Managing and monitoring the use of available disk space, memory, connection error logs, state of transaction logs, device problems, etc.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to the creation of new database devices, the allocation of appropriate disk space, and maintenance of database segments.

Specific Objective 2 - The student will be able to start and shutdown the SQL server.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating the startup and shutdown of the SQL server.

Specific Objective 3 - The student will be able to perform database user account and access privilege procedures including:

- Creating user accounts.

- Granting and revoking access privileges for data retrieval, insertion, deletion and update of objects.
- Granting and revoking roles for SQL server users groups.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to the creation of user accounts and the granting and revoking of access privileges.

Specific Objective 4 - The student will be able to perform database security and auditing procedures.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database security and auditing procedures.

Specific Objective 5 - The student will be able to perform database integrity monitoring.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database integrity monitoring.

Specific Objective 6 - The student will be able to perform database backups on a regular or on-demand basis.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database backups.

Specific Objective 7 - The student will be able to perform a recovery of the database following a system failure or on-demand.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database recovery.

Specific Objective 8 - The student will be able to configure databases unique to ECS DAACs including:

- Making database size estimates and planning.
- Preparing database-unique attributes.
- Preparing database reports.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database configuration specific to the ECS DAACs.

Specific Objective 9 - The student will be able to perform database tuning and performance monitoring procedures including:

- Design and indexing.
- Responding to queries.
- Monitoring and boosting performance.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database tuning and performance monitoring.

Specific Objective 10 - The student will be able to maintain the interface file to the SQL server.

Condition - The student will be given a copy of *625-CD-011-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to maintaining the interface file to the SQL server.

Importance

ECS relies on vast amounts of data from the science user's perspective and from a maintenance and operations perspective. Accurate information stored in the science databases allows science user's to access necessary data quickly. Similarly, databases that maintain operational data must be kept current in order for routine and specialized administrative tasks to be performed.

Special Instructions on Procedures

All procedures in this training guide assume that you are logged in to a system server or workstation as yourself.

Many DBA tasks will be performed by using scripts that are run from the command line. Some scripts may require modification which will require you to use a text editor of your choice. These procedures do not give keystroke-by-keystroke information on how to edit the file; they only instruct you to make the changes required to perform the task.

This page intentionally left blank.

Overview

Database administration involves the maintenance of databases by installing SQL server products, managing disk storage space, managing user accounts and privileges, and performing database backup and recovery operations.

DBA Tasks and Functions

The Database Administrator (DBA), is the individual or office responsible for the installation, configuration, update, maintenance, and overall performance and reliability of the SQL Server database. In general, the DBA is concerned with the availability of the server, the definition and management of resources allocated to the server, the definition and management of databases and objects resident on the server, and the relationship between the server and the operating system.

The DAAC Database Administrators perform the following functions:

- Perform the database administration utilities, such as database backup, maintenance of database transaction logs, and database recovery.
- Monitor and tune the physical allocation of database resources.
- Maintain user accounts.
- Create user registration and account access control permissions in the security database.
- Work with data specialists in information management tasks involving databases, data sets, and metadata management.
- Perform daily database synchronization.

Sybase Directory Structure

The following table describes the directory structure used for Sybase database administration in Version 2.0.

\$SYBASE is a shortcut for the full path to the Sybase home directory. Because each DAAC may support a different full path to that directory (e.g., /usr/ecs/OPS/COTS/sybase, /vol0/sybase, etc.), this document will refer to the path as **\$SYBASE**.

Table 1. Sybase Directory Structure.

Directory	Contains
\$SYBASE/bin	Utilities necessary to load, run, and access the server
\$SYBASE /install	Files used to start dataservers, backupserver and to record server messages (errorlogs)
\$SYBASE /scripts	Root directory for all script files executed on the server
\$SYBASE/scripts/mk_DB_database.sql (e.g. database = SubServer)	SQL script files used to create databases on the server
\$SYBASE/scripts/add_devices.sql	SQL script file (disk init) used to map physical storage to a logical name.
\$SYBASE/scripts/alter_DB_tempdb.sql	SQL script file used to alter the database.
\$SYBASE /scripts/del_user	Script file used to delete logins and remove them from the users database.
backup directory: \$SYBASE/sybase.dumps	Root directory that contains all backup subdirectories. <i>It is recommended, but not required, that this directory be on a separate physical disk</i>
backup subdirectory: \$SYBASE/sybase.dumps/dumpdir	Created each evening by the RUN_backups cron job, they are named DATABASE_NAME.dat.Z. (e.g. DATABASE_NAME = SubServer)
/tools/syb0Cv11.1.0	HP's and SUN's db-lib, ct-lib, and xa-lib client library files used by applications to gain access to the server
/tools/syb0Cv11.1_32	SGI's db-lib, ct-lib, and xa-lib client library files used by applications to gain access to the server

Naming Conventions

In order to keep track of the large number of files that you will create and work with throughout your tenure as DBA, and because other people will come into contact with these files at one time or another, please follow these simple naming conventions for your files:

- The file name should indicate the function and/or content of the object regardless of the length of the file name.
- Only easily understandable abbreviations should be used.
- Parts of names are separated by the underscore character (_).
- Only one optional suffix is permitted and is appended to the file name by a period (.).
- The full path of the object is considered to be part of the name.

Some examples of these files are as follows:

Version 2.0 has eleven SQL Servers installed: 1 APC Server (**t1acg01_srvr**), 1 Primary Ingest Server (**t1icg01_srvr**), 1 Secondary Ingest Server (**t1icg03_srvr**), 1 Primary Application Server (**t1mss06_srvr**), 1 Secondary Application Server (**t1mss07_srvr**), 1 Primary Interface Server (**t1ins01_srvr**), 1 Secondary Interface Server (**t1ins02_srvr**), 1 Planning Server (**t1pls01_srvr**), 1 Queuing Server (**t1sps02_srvr**), 1 MSS Primary Server (**t1msh01_srvr**) and 1 Fail-Over MSS Primary Server (**t1msh08_srvr**).

All backup files are located under the \$SYBASE/sybase.dumps/dumpdir directory, which may be on a separate physical disk. These files are kept for a period of 1 day and they are named **database_name.dat.Z** where **database_name** is the database name for that server. For example, a backup of **IoAdAdvService** database would be in a backup file called **IoAdAdvService.dat.Z**.

All SQL script files have the **.sql** suffix. Their names reference the objects they create or functions they perform, and are all located either in \$SYBASE/scripts or below. A SQL file to create the **IoAdAdvService** database is named **\$SYBASE/scripts/mk_DB_IoAdAdvService.sql**, and another file used to alter the database might be called **\$SYBASE/scripts/alter_DB_tempdb.sql**.

This page intentionally left blank.

Version 2.0 Databases

Version 2.0 Databases

Version 2.0 databases are divided into two categories:

- Production databases contain data used in the generation of end-user products such as the Earth Science Data Types (ESDTs), Product Generation Executables (PGEs), and metadata.
- System Management databases hold results of system activity and access, such as configuration management, network activity, and trouble tickets.

Table 2 and Table 3 below detail the information stored in each database.

Table 2. Version 2.0 Production Databases

Database	Content
Ingest DB	File types to be ingested, status of ingest production requests, historical summary.
Planning & Data Processing System (PDPS)	PGE information to plan production.
Metadata	Collection, granule, document, producer, and other descriptive information about science data.
MSS Management DB	System and application performance information, user profiles, and order tracking information.
Storage Management Pull Monitor	Staging disk usage, tape drives available, requests needing resources.
Data Dictionary	User descriptions of attributes and other terms in system
Document Database (WEB-based search only)	ECS-related science documents.
ASTER Lookup Table (LUT)	Atmospheric correction parameters
CSS Persistent Access Control List (ACL)	List of principals who can access components.
Advertising Database	Holdings and services available to users.
Subscription Database	Principals (users or servers) to be notified when a specified event occurs.

Table 3. Version 2.0 System Management Databases

Application	Tool	Server
Network Management	HP OpenView	MSS
System Performance Management	Tivoli TME/Sentry	MSS
Extensible SNMP Agent	Peer Networks Optima	All platforms
Trouble Ticketing	Remedy Corporation ARS	MSS
Physical Configuration Management	Accugraph Corporation (PNM)	MSS
Security/DCE Management	HAL DCE Cell Manager	CSS
Software Change Management	Clearcase	CM
Change Request Management	DDTS	CM
Baseline Manager	HTG XRP	CM

Maintaining the Interface File

An SQL Server entry in the interfaces file is required for you to connect to a SQL Server. There can be more than one SQL Server entry in the interfaces file. An entry in the interfaces file lets you connect to a SQL Server not physically on that machine.

Adding an entry to the interfaces file

Adding an entry to the interfaces file (When a new SQL Server is created):

- 1 Type, *su – sybase* and press **Return**.
- 2 Type, *cd \$SYBASE/install* and press **Return**.
- 3 (where \$SYBASE is the directory where the Sybase software resides or the Sybase Open Client)
- 4 Type *./sybinit* and press **Return**
- 5 Type, *1* (Release directory) and press **Return**
- 6 If it is correct, the press **Return**. If not correct, then type the correct path and press **Return**
- 7 Type, *2* (Edit / View Interfaces File) and press **Return**
- 8 Type, *1* (Add a new entry) and press **Return**
- 9 Type, *1* (Server Name) and press **Return**
- 10 Type the *servername* (e.g. g0mss01_srvr) and press **Return**
- 11 Press, *Ctrl-A*
- 12 Type, *1* (Retry Count) and press **Return**
- 13 Type, *5* and press **Return**
- 14 Type, *2* (Retry Delay) and press **Return**
- 15 Type, *5* and press **Return**
- 16 Type, *3* (Add a new listener service) and press **Return**
- 17 Type, *1* (Hostname/Address) and press **Return**
- 18 Here, you type the host name of the machine where the SQL Server resides. If the hostname is correct, press **Return**. If it is not correct, then type the correct host name and press **Return**
- 19 Type, *2* (Port) and press **Return**
- 20 Here, you type the port number on which the SQL Server is running or will be running and press **Return**
- 21 Press, *Ctrl-A*

- 22 Type “y”, when you are prompted with “Is this information correct?”
 - 23 Press, **Ctrl-A**
 - 24 Type “y”, when you are prompted with “Write the changes to the interfaces file now?”
 - 25 Press, **Ctrl-A**
 - 26 Press, **Ctrl-A**
-

Modifying an existing entry in the interfaces file

Modifying an existing entry in the interfaces file

(When the machine that has the SQL Server is running gets moved):

- 1 Type, *su – Sybase* and press **Return**.
 - 2 Type, *cd \$SYBASE/install* and press **Return**.
 - 3 (where \$SYBASE is the directory where the Sybase software resides or Sybase Open Client)
 - 4 Type, *./sybinit* and press **Return**.
 - 5 Type, *1* (Release directory) and press **Return**
 - 6 If it is correct, the press **Return**. If not correct, then type the correct path and press **Return**
 - 7 Type, *2* (Edit / View Interfaces File) and press **Return**
 - 8 Type, *2* (Modify an existing entry) and press **Return**
 - 9 Type, in the number for the SQL server you are modifying (The number is on the right hand side of the SQL server entry and press **Return**
 - 10 Press, **Ctrl-A**
 - 11 Type, *4* and press **Return**. This is the only entry you might have to change.
 - 12 Type, *1* (Hostname/Address) and press **Return**
 - 13 Type, the *correct hostname or IP address* and press **Return**
 - 14 Type, “y”, when you are prompted with “Is this information correct?”
 - 15 Press, **Ctrl-A**
 - 16 Type, “y”, when you are prompted with “Write the changes to the interfaces file now?”
 - 17 Press, **Ctrl-A**
 - 18 Press, **Ctrl-A**
-

Deleting an existing entry in the interfaces file:

- 1 Type, *su – Sybase* and press **Return**.
 - 2 Type, *cd \$SYBASE/install* and press **Return**.
 - 3 (where **\$SYBASE** is the directory where the Sybase software resides or Sybase Open Client)
 - 4 Type, *./sybinit* and press **Return**.
 - 5 Type, *I* (Release directory) and press **Return**.
 - 6 If it is correct, the press **Return**. If not correct, then type the correct path and press **Return**.
 - 7 Type, in the *number for the SQL server you want to delete* (The number is on the right hand side of the SQL server entry and press **Return**.
 - 8 Type, “y”, when you are prompted with “Remove this entry from the interfaces file?”
 - 9 Type, “y”, when you are prompted with “Write the changes to the interfaces file now?”
 - 10 Press, *Ctrl-A*
 - 11 Press, *Ctrl-A*
-

This page intentionally left blank.

Starting and Stopping SQL Server

Start an SQL Server Process

A SQL server process is started when a new server is installed or after a system outage.

In order to perform the procedure, the DBA must have obtained the database server name.

To start a SQL server process, the DBA must have sa_role (SQL Server)

Start the SQL Server Process Procedure

- 1 Log into the server on which the new user database is to be created by typing: **telnet *ServerName*** or **rlogin *ServerName*** or **rsh *ServerName***, then press **Return**.
 - If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
 - If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.
 - 2 If a **Login:** prompt appears, log in as yourself by typing: ***sybase***, then press **Return**.
 - A password prompt is displayed.
 - 3 Enter ***sybase password***, then press **Return**.
 - Remember that ***sybase Password*** is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - 4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber:0.0*** or **setenv DISPLAY *ServerName:0.0***, then press **Return**.
 - 5 At the UNIX prompt, type, **cd install** and then press **Return**.
 - 6 At the UNIX prompt, type **./RUN_*servername* &)**, then press **Return**.
 - 7 At the UNIX prompt, type **showserver**, then press **Return**.
 - This displays a listing of the SQL Server processes that are running.
-

Stop an SQL Server

A SQL server process is stopped by the DBA when the system to be brought down for maintenance. The **shutdown** command issued from within ISQL shuts down the SQL server on which you are logged in. It allows for the completion of any current SQL processes and blocks the start of any new SQL processes before the server is shutdown. Adding a **nowait** option to

the command immediately terminates all processes and shuts down SQL server. Using the **nowait** option may result in loss of data and a more complicated recovery procedure, so use it sparingly.

Stop the SQL Server Procedure

- 1 Log into the server on which the new user database is to be created by typing: **telnet *ServerName*** or **rlogin *ServerName*** or **rsh *ServerName***, then press **Return**.
 - If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
 - If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.
 - 2 If a **Login:** prompt appears, log in as yourself by typing: **sybase**, then press **Return**.
 - A password prompt is displayed.
 - 3 Enter **SybasePassword**, then press **Return**.
 - Remember that **SybasePassword** is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.
 - 4 Set display to current terminal by typing: **setenv DISPLAY IPNumber:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.
 - 5 Type, **setenv SYBASE sybase-directory-path**
 - 6 Type, **setenv DSQUERY sql-servername**
 - 7 Type, **\$SYBASE/bin/isql -Usa** and press **Return**.
 - 8 Type, the **sa-password** when prompted for it and press **Return**.
 - 9 At the ISQL prompt, type, **shutdown with nowait** at “1>” prompt and press **Return**; type **go** at “2>”, then press **Return**.
 - 10 At the ISQL prompt, type **showserver**, then press **Return**; type **go**, then press **Return**.
 - This displays a listing of the SQL Server processes that are running.
 - There should be no SQL Server processes running.
 - 11 At the ISQL prompt, type **exit**, then press **Return**.
 - You are returned to a UNIX prompt.
-

Starting the SQL Backup Server Procedure

(NOTE: This step should be done after SQL Server is up and running)

-
- 1 Login into the UNIX machine on which you want to start the SQL Server by typing: **telnet *ServerName***, and press RETURN
 - 2 When the **Login:** prompt appears, log in by typing: **sybase** and press RETURN
 - 3 When the **Password:** prompt appears, type ***sybase password*** and press RETURN
 - 4 Type, **cd install** and press RETURN
 - 5 Type, **./RUN_backup-servername &** and press RETURN
-

Stopping the SQL Backup Server Procedure

(NOTE: This step should be done before shutting down the SQL Server)

-
- 1 Login into the UNIX machine on which you want to start the SQL Server by typing: **telnet *ServerName***, and press RETURN
 - 2 When the **Login:** prompt appears, log in by typing: **sybase** and press RETURN
 - 3 When the **Password:** prompt appears, type ***sybase password*** and press RETURN
 - 4 Type, **setenv SYBASE *sybase-directory-path***
 - 5 Type, **setenv DSQUERY *sql-servername***
 - 6 Type, **\$SYBASE/bin/isql -Usa** and press RETURN
 - 7 Type, the ***sa-password*** when prompted for it and press RETURN
 - 8 Type, **shutdown SYB_BACKUP** at “1>” prompt and press RETURN
 - 9 Type, **go** at “2>” prompt and press RETURN
-

Starting the SQL Monitor Server Procedure

(NOTE: This step should be done after the SQL Server is up and running)

-
- 1 Login into the UNIX machine on which you want to start the SQL Server by typing: **telnet *ServerName***, and press RETURN
 - 2 When the **Login:** prompt appears, log in by typing: **sybase** and press RETURN
 - 3 When the **Password:** prompt appears, type ***sybase password*** and press RETURN
 - 4 Type, **cd SMS11.x/bin** and press RETURN
 - 5 Type, **./monserver -S*server-name* -M*monitor-server-name* -Usa -P*sa-password* &** and press RETURN
-

NOTE: As you can see, this is not a good way of starting Monitor Server, because you are actually typing the sa-password and if someone does a UNIX “ps -ef” command, they can see the sa-password. A startup script is a better way of starting the Monitor Server.

Stopping the SQL Monitor Server Procedure

(NOTE: This step should be done before the SQL Server is shutdown)

- 1 Login into the UNIX machine on which you want to start the SQL Server by typing: **telnet *ServerName***, and press RETURN
 - 2 When the **Login:** prompt appears, log in by typing: **sybase** and press RETURN
 - 3 When the **Password:** prompt appears, type ***sybase password*** and press RETURN
 - 4 Type, **setenv SYBASE *sybase-directory-path***
 - 5 Type, **setenv DSQUERY *sql-servername***
 - 6 Type, **\$SYBASE/bin/isql -Usa** and press RETURN
 - 7 Type, the ***sa-password*** when prompted for it and press RETURN
 - 8 Type, **sms_shutdown** at “1>” prompt and press RETURN
 - 9 Type, **go** at “2>” prompt and press RETURN
-

Database Devices

A database device stores the objects that make up databases. The term *device* does not necessarily refer to a distinct physical device; it can refer to any piece of disk, such as a partition; or a file in the file system used to store databases and their objects (Figure 1).

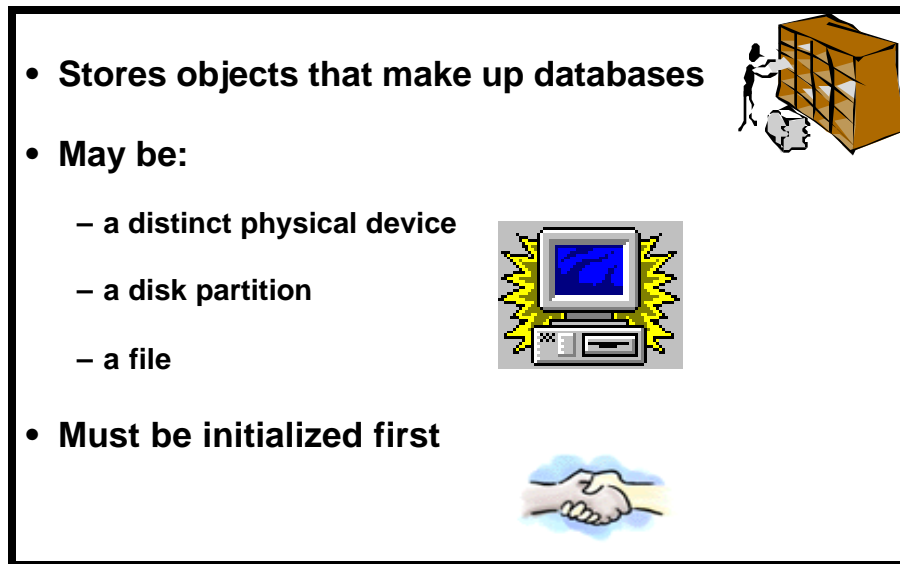


Figure 1. Database Devices

Each database device or file must be prepared and made known to the SQL Server before it can be used for database storage, the process is called initialization. Once a database device is initialized, it can be:

- Allocated to the pool of space available to a user database.
- Allocated to a user database, and assigned to store a specific database object or objects.
- Used to store a database's transaction logs.
- Designated as a default device for **create** and **alter** database commands.

A database device is created when the System Administrator determines that new disk space is available for use by a Sybase database, or as part of the Database Recovery Procedure. The System Administrator makes a request to the DBA who creates the new database device and notifies the System Administrator when the device has been created.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database device to create.

- Physical device on which to place database device.
- Device size in megabytes.
- For a mirrored device, name of the mirror device.

Making Database Size Estimates and Planning

Before a database is created, a user has to estimate the size of the database he requires based on how much data is thinks he would be stored. This estimate will be the data size. By default, the log size will be 1/5th the data size. While estimating the data size, a user has to keep in mind the future growth.

Preparing Database Unique Attributes (after a database has been created):

Setting the right database options:

Command: **sp_dboption *database-name*, *option-name*, true**

Setting a Data Base Owner (dbo):

Command: **sp_changedbowner *user-name***

(It is assumed that the user already has a login in the SQL Server and you are currently in the database you want to change the dbo of, as an “sa”)

Create Database Device Procedure

-
- 1 Log into the server on which the new database devices it to be created by typing: **telnet *ServerName*** or **rlogin *ServerName*** or **rsh *ServerName***, then press **Return**.
 - If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
 - If you use **rlogin** or **rsh**, the system uses the same logname currently in use and prompts you for a password. Skip to Step 3 below.
 - 2 If a **Login:** prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.
 - A password prompt is displayed.
 - 3 Enter ***YourPassword***, then press **Return**.
 - Remember that ***YourPassword*** is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - 4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber:0.0*** or **setenv DISPLAY *ServerName:0.0***, then press **Return**.

For each database device to be created, perform Steps 5 through 9:

- 5 Log into Sybase by typing: **su sybase**, then press **Return**.
 - A password prompt is displayed.
- 6 Enter **SybasePassword**, then press **Return**.
 - Remember that **SybasePassword** is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.
- 7 At the UNIX prompt, type **cd scripts**, then press **Return**.
 - This places you in the directory that contains all the scripts used to create database devices.
 - It is important that these scripts not be deleted from the system in case it is necessary to restore a database device following a failure (see *Restore Database Devices*).
- 8 At the UNIX prompt, type **cp add_devices.sql DeviceName.sql**, then press **Return**.
 - This creates a new SQL script file into which you will type the appropriate commands to create the database device.
- 9 Using the text editor of your choice, edit **DeviceName.sql** (Figure 2) and make changes to information enclosed in brackets (including the brackets) as appropriate:

```
/* **** */
/* name: [add_devices.sql] */
/* purpose: */
/* written: */
/* revised: */
/* reason: */
/* **** */
disk init name = [device name],
physname = "/dev/[device name]",
vdevno = [#],
size = [size]
```

Figure 2. add_devices.sql file for creation of a database device.

- In the area delimited by **/* */**, enter an appropriate description of the script including the file name, date written and person who wrote the script, it's purpose, and any other

information deemed appropriate. Be sure to enclose each line of the comment between */* */*.

- The **disk init name** is the *DeviceName* is used in Step 8 above.
 - You may not use spaces or punctuation except the underscore character (*_*) in *DeviceName*.
 - Remember that the name you assign is case sensitive.
 - Be sure there is a comma after *DeviceName*
- The **physname** is the *FullPath_to_DeviceName..*
 - Be sure to enclose *FullPath_to_DeviceName* in double quotes.
 - Be sure to place a comma after *FullPath_to_DeviceName* but outside the double quotes.
- The size is the *DeviceSize* in blocks.
 - To compute the number of blocks, multiply the device size in megabytes by 512; e.g., a 1,000 Mb device has 512,000 blocks
 - Be sure to place a comma after *DeviceSize*.
- The **vdevno** is the *VirtualDeviceNumber*
 - **vdevno** is a unique identifying number for the database device. It must be unique among the all devices used by SQL Server and is never reused. Device number 0 represents the device named **d_master** that stores the system catalogs. Legal numbers are between 1 and 255, but the highest number must be one less than the number of database devices for which your system is configured. For example, for a system with the default configuration of 10 devices, the legal device numbers are 1-9. The next available number can be determined by looking at the output from the **sp_helpdevice** command and selecting the next number in sequence. If you use an existing number, Sybase will return the message, “device activation error.”

10 After the changes have been made, save the file according to the rules of your text editor.

11 At the UNIX prompt, type:

isql -Usa -SServerName -iDeviceName.sql -oDeviceName.rpt

then press **Return**.

- *ServerName* is the name of the server on which the database device will be created.
- *DeviceName.sql* is the name of the script file you created in step 8.
- *DeviceName.rpt* is the filename of the script’s output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

12 At the **Password:** prompt, type the *SybaseSAPassword*, then press **Return**.

13 When the UNIX prompt is again displayed the process is complete.

14 At the UNIX prompt, type **more *DeviceName.rpt*** , then press **Return**.

- This allows you to view the *DeviceName.rpt* file to confirm that the device has been created or to check for device creation errors.
-

A sample of a completed Database Device creation script follows (Figure 3).

```
/* **** */
/* name: subserver.sql */
/* purpose: tracking training program */
/* written: */
/* **** */
disk init name = subserver_data,
physname = "/dev/vx/rdsk/rootdg/SubServer_Data",
vdevno = 3,
Size = 1024000
```

Figure 3. Completed database device creation script.

This page intentionally left blank.

User Databases

User Databases

A user database is created when an approved request is received by the DBA, or as part of the Database Recovery Procedure (see **Database Recovery**). The database name, approximate size of the database and the transaction log are necessary information for the DBA to complete the task of creating a user database. It is the user's responsibility to determine the appropriate database design.

The requester fills out a "User Database Request Form" and submits it to the supervisor.

The requester's supervisor reviews the request, and if s/he determines that it is appropriate to create the User Database, forwards the request to the Operations Supervisor (Ops Super). The Ops Super verifies that all required information is contained on the form. (Incomplete forms are returned to the requester's supervisor for additional information.) If it is complete and if the request for a new User Database fits within policy guidelines, the Ops Super approves the request and forwards the request form to the DBA to implement. After the User Database is created, the DBA notifies the requester that the new database is available. The DBA also sends an e-mail message to the user's supervisor informing them that the new User Database was created.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database to create.
- Size of database and the transaction log.
- Database devices to use.
- To create a database, the DBA must have sa_role (SQL Server) privileges.

Create User Database Procedure

- 1 Log into the server on which the new user database is to be created by typing: **telnet *ServerName*** or **rlogin *ServerName*** or **rsh *ServerName***, then press **Return**.
 - If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
 - If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.
- 2 If a **Login:** prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.
 - A password prompt is displayed.
- 3 Enter ***YourPassword***, then press **Return**.
 - Remember that ***YourPassword*** is case sensitive.

- You are authenticated as yourself and returned to the UNIX prompt.
- 4 Set display to current terminal by typing: **setenv DISPLAY IPNumber:0.0** or **setenv DISPLAY ServerName:0.0**, then press **Return**.
- 5 Log into Sybase by typing: **su sybase**, then press **Return**.
- A password prompt is displayed.
- 6 Enter *SybasePassword*, then press **Return**.
- Remember that *SybasePassword* is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.
- 7 At the UNIX prompt, type **cd scripts/**, then press **Return**.
- This places you in the directory that contains all the scripts used to create database devices.
 - It is important that these scripts not be deleted from the system in case it is necessary to restore a database device following a failure (see **Restore Databases**).
- 8 At the UNIX prompt, type **cp mk_DB_database.sql DBName.sql**, then press **Return**.
- This creates a new SQL script file into which you will type the appropriate commands to create the user database.
- 9 Using the text editor of your choice, edit **DBName.sql** (Figure 4) and make changes to information enclosed in brackets (including the brackets) as appropriate:
- **DBName** is the name of the database and should describe it's function succinctly.
 - **DBDeviceName** is the name of the existing, approved database device on which **DBName** will reside.
 - **DBSize_in_MB** is the estimated size of the database in megabytes.
 - **LogDBDeviceName** is the name of the database device holding the transaction log
 - **LogSize_in_MB** is the estimated size of the transaction log in megabytes.
 - The **sp_helpdb** command provides feedback at the end of the script to assure that the database has been created.

```

/*****
/* name: [mk_DB_database-name.sql]          */
/* purpose:                                  */
/* written:                                  */
*****/
create database [DBName]
on [DBDeviceName] = [DBSize_in_MB]
log on [LogDBDeviceName] = [LogSize_in_MB]
go
sp_helpdb [DBName]
go

```

Figure 4. Sample template.sql file for creation of a database.

10 After the changes have been made, save the file according to the rules of the text editor.

11 At the UNIX prompt, type:

isql -Usa -SServerName -iDBName.sql -oDBName.out

then press **Return**.

- **ServerName** is the name of the server on which the database device will be created.
- **DBName.sql** is the name of the script file you created in step 9.
- **DBName.out** is the filename of the script's output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

12 At the **Password:** prompt, type the **SybaseSAPassword**, then press **Return**.

- Remember the **SybaseSAPassword** is case sensitive.

13 When the UNIX prompt is again displayed the process is complete.

14 At the UNIX prompt, type **more DBName.out** , then press **Return**.

- This allows you to view the **DBName.out** file to confirm that the device has been created or to check for database creation errors.

A sample of a completed Create Database script follows (Figure 5).

```

/*****
/* name: mk_DB_SubServer.sql      */
/* purpose: track specific data items */
/* written:                      */
*****/

create database SubServer
on subserver_data = 100, subserver_index=50
log on subserver_log = 50
go
sp_addlogin css_role, welcome, SubServer
go
sp_addlogin subServerApp, welcome, SubServer
go
use SubServer
sp_changedbowner css_role
go
checkpoint
go
use_master
go
etc..

```

Figure 5. Completed Create DatabaseSscript.

Renaming Database:

Command:

- 1 At the UNIX prompt, type, then press **Return**.

isql -Usa

- The system will prompt you for a password.

- 2 At the **Password:** prompt, type the *SybaseSAPassword*, then press **Return**.

- Remember the *SybaseSAPassword* is case sensitive.

- 3 Type the following then press **Return**.

/ Rename database Old-database-name to New-database-name */*

- 4 At the Sybase prompt, type the following:

```
sp_dboption Old-database-name, "single user", true
go
use Old-database-name
go
checkpoint
go
use master
go
sp_renamedb Old-database-name, New-database-name
go
sp_dboption New-database-name, "single user", false
go
use New-database-name
go
checkpoint
go
use master
go
```

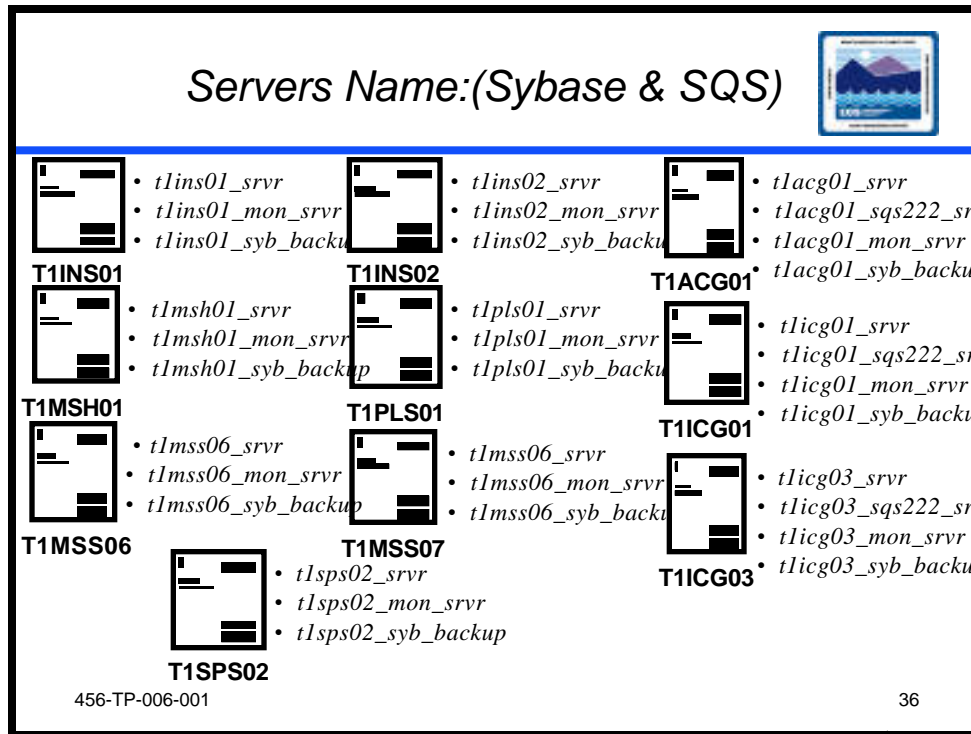


Figure 5a. VATC: Server names (Sybase & SQS).

Changing Password:

Command:

- 1 At the UNIX prompt, type:

isql -Usa

then press **Return**.
The system will prompt you for a password.
- 2 At the **Password:** prompt, type the **SybaseSAPassword**, then press **Return**.
Remember the **SybaseSAPassword** is case sensitive.
- 3 When the Sybase prompt, type the following:

sp_password old-password, new-password, user-name

Database Segments

Creating Segments

A **segment** is a collection of the database devices and/or fragments available to a particular database. Tables and indexes can be assigned to a particular segment, and thereby to a particular physical device, or can span a set of physical devices.

Segments are named subsets of the database devices available to a particular SQL Server database. A segment can best be described as a label that points to one or more database devices. Segment names are used in **create table** and **create index** commands to place tables or indexes on specific database devices. The use of segments can increase SQL Server performance, and can give the System Administrator or Database Owner increased control over placement, size and space usage of specific database objects. For example:

- If a table is placed on one device, and its non-clustered indexes on a device on another disk controller, the time required to read or write to the disk can be reduced, since disk head travel is usually reduced.
- If a large, heavily-used table is split across devices on two separate disk controllers, read/write time may be improved.

SQL Server stores the data for text and image columns on a separate chain of data pages. By default, this text chain is placed on the same segment as the table. Since reading a text column requires a read operation for the text pointer in the base table and an additional read operation on the text page in the separate text chain, placing the text chain on a separate physical device can improve performance.

If you place tables and indexes only on specific segments, those database objects cannot grow beyond the space available to the devices represented by those segments, and other objects cannot contend for space with them.

Segments can be extended to include additional devices as needed.

You can use thresholds to warn you when space becomes low on a particular database segment.

Segments are created within a particular database from the database devices already allocated to that database. Each SQL Server database can contain up to 32 segments. The database devices must first be initialized with **disk init**, and then be made available to the database with a **create database** or **alter database** statement before you can assign segment names.

When you first create a database, SQL Server creates three segments in the database (Table 4):

Table 4. Default segment names and functions.

Segment	Function
system	Stores this database's system tables.
logsegment	Stores this database's transaction log.
default	Stores all other database objects-unless you create additional segments, and store the table or index on the new segments by using create table... on segment_name or create index... on segment_name .

Database segments are created when a database is created, when the Database Administrator determines that it is necessary to allocate database objects to a new or additional devices, or as part of the Database Recovery Procedure.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database
- Database device extents
- Space on the database allocated to the Database device

To create a database, the DBA must have sa_role.

Create Database Segment Scenario and Procedure

The DBA receives a request to create a segment for the storage of the SubServer table indexes in the t1ins01_srvr database, on a separate physical disk. Two devices **subserver_data** and **subserver_index** have already been created and are located on different physical disks.

- 1 At the UNIX prompt, type **cd /scripts**, then press **Return**.
- 2 Using the text editor of your choice, edit **segment.sql** (Figure 6) and make changes to information enclosed in brackets (including the brackets) as appropriate:

Note: This file doesn't exist. It must be created.

```
/* ****  
/* name: [segment.sql]  
/* purpose:  
/* written:  
/* revised:  
/* reason:  
/* ****  
  
sp_addsegment [seg_name], [DBname],
```

Figure 6. Create database segment template file.

- 3 After the changes have been made, save the file according to the rules of the text editor.
- 4 At the UNIX prompt, type: **isql -Usa -Sservername -iSegment.sql -oSegment.out** then press **Return**.
 - The system will prompt you for a password.
- 5 At the **Password:** prompt, type the **SybaseSAPassword**, then press **Return**.

- Remember the *SybaseSAPassword* is case sensitive.
- 6** When the UNIX prompt is again displayed the process is complete.
- 7** At the UNIX prompt, type **more *Segment.out*** , then press **Return**.
- This allows you to view the *Segment.out* file to confirm that the device has been created or to check for database creation errors.
-

This page intentionally left blank.

Transaction Logs

What is a Transaction?

A **transaction** is a the SQL Server's basic unit of work. Each SQL statement that modifies data in a database is considered a transaction, and SQL Server uses them to keep track of all database changes. A transaction consists of one or more Transact-SQL statements that succeed or fail as a unit. Each success or failure is recorded automatically in the database's transaction log.

What is a Transaction Log?

The Transaction Log is an ever-expanding file that records each and every transaction that occurs in a database. This log is used to perform a complete recovery of the database in the event of a media failure. The transaction log is normally maintained on a different database device or even a completely separate system than the rest of the database so that it can be read should a failure occur.

How Transaction Logging Works

The transaction log is a write-ahead log. When a user issues a statement that would modify the database, SQL Server automatically opens the transaction log and writes a **transaction start** statement. After all changes for a statement have been recorded, a **transaction end** statement is written to the log, and the log is written to an in-cache copy of the data page. The data page remains in cache until the memory is needed for another database page at which time the **checkpoint** process writes the changes to disk.

If any statement in a transaction fails to complete due to operator error, the operator aborting the procedure, or mechanical failure, SQL Server automatically reverses all changes made by the transaction. SQL Server writes an "end transaction" record to the log at the end of each transaction, recording the status (success or failure) of the transaction.

Each database has its own transaction log that records all changes to its database which may or may not be on the same database device. It is strongly recommended that transaction logs be kept on a device separate from the database as insurance against catastrophic data loss.

The transaction log grows in size over time and may cause database problems when full. Therefore, the transaction log must be dumped periodically.

Getting Information about the Transaction Log

The transaction log contains enough information about each transaction to ensure that it can be recovered. Use the **dump transaction** command to copy the information it contains to tape or disk. Use **sp_spaceused syslogs** to check the size of the log, or **sp_helpsegment logsegment** to check the space available for log growth.

Getting Information About The Transaction Log

- 1 At the UNIX prompt, type:
isql -Usa
then press **Return**.
The system will prompt you for a password.
- 2 At the **Password:** prompt, type the *SybaseSAPassword*, then press **Return**.
Remember the *SybaseSAPassword* is case sensitive.
- 3 When the Sybase prompt, type **sp_helpdb**, the press **Return**,
type **go**, then press **Return**.
- 4 Select the database that you want to view,
type **sp_helpdb SubServer**, the press **Return**.
type **go**, then press **Return**.

Output:

<u>device_fragments</u>	<u>size</u>	<u>usage</u>	<u>free kbytes</u>
subserver_log	50.0 MB	log only	51056

- 5 At the SYBASE prompt, type **exit**, then press **Return**.
-

Backup and Recovery

Database and transaction log backups and recoveries are probably the most important tasks the Database Administrator must perform. Without regular and complete backups of databases and transaction logs, the potential for enormous amounts of data to be lost is very great.

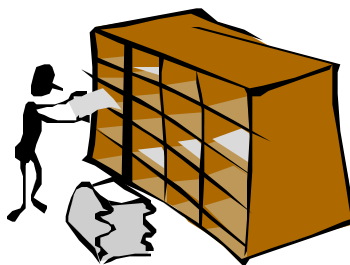
Frequency and Schedule

Each DAAC is responsible for determining its own backup schedule to meet its individual requirements. Nonetheless, it is strongly suggested that a regular schedule of database backups be established and maintained. Although the databases are included in the daily backups of the entire system, separate database backups should be performed nightly.

Database backups are run by a UNIX **cron** job which executes the **sybasedump** program located in the **\$SYBASE/sybase.dumps** directory. No intervention in the automatic backup process is required by the DBA, though periodic checks of the Backup subdirectories are recommended.

Manual backups can be performed at any time by the DBA and are recommended for the following situations:

- Any change to the **master** database, including new logins, devices, and databases.
- Any major change to user databases, such as a large ingest or deletion of data, definition of indexes, etc.,
- Other mission-critical activities as defined by the DAAC operations controller.



Manual Database Backup Procedure

- 1 Log into the server on which contains the database to be backed up by typing: **telnet *ServerName*** or **rlogin *ServerName*** or **rsh *ServerName***, then press **Return**.
 - If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.

- If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.
- 2 If a **Login:** prompt appears, log in as yourself by typing: *YourUserID*, then press **Return**.
 - A password prompt is displayed.
 - 3 Enter *YourPassword*, then press **Return**.
 - Remember that *YourPassword* is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - 4 Set display to current terminal by typing: **setenv DISPLAY IPNumber:0.0** or **setenv DISPLAY ServerName:0.0**, then press **Return**.

For each database device to be created, perform Steps 5 through yyy:
 - 5 Log into Sybase by typing: **su sybase**, then press **Return**.
 - A password prompt is displayed.
 - 6 Enter *SybasePassword*, then press **Return**.
 - Remember that *SybasePassword* is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.
 - 7 At the UNIX prompt, type **isql -Usa**, then press **Return**.
 - 8 To backup the database, at the ISQL prompt, type **dump database DBName to BackupDirectory/DBName.dat**, then press **Return**; type **go**, then press **Return**.
 - 9 To backup the transaction log, at the ISQL prompt, type **dump transaction DBName to BackupDirectory/DBName_tx.dat**, then press **Return**; type **go**, then press **Return**.
-

Database Recovery

A database recovery is performed when the Database Administrator determines that some database data is corrupt, or when the System Administrator determines that a database device has failed. The Database Administrator makes a request to the System Administrator, who performs the restore and notifies the Database Administrator when the restore is complete.

The symptoms of media failure are as variable as the causes. If only a single block on the disk is bad, your database may appear to function perfectly for some time after the corruption appears; this is why you should run **dbcc** commands frequently. If an entire disk or disk controller is bad, you will not be able to use a database. SQL Server marks it as suspect and displays a warning message. If the disk storing the master database fails, users will not be able to log into the server,

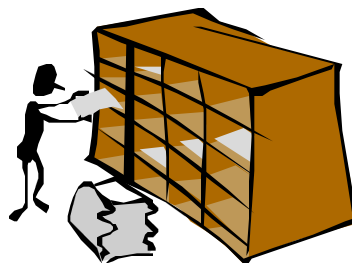
and users already logged in will not be able to perform any actions that access the system tables in master.

The complete database device restoration procedure is actually a suite of procedures that must be performed in the prescribed order:

1. The device failure has been verified by the System Administrator and requests restoration from the DBA.
2. If possible, perform a dump of the current database and the current database transaction log for each database on the failed device will be backed up. If this is not possible due to the damage to the database device, then the DBA will have to use the most recent database and transaction log dumps.
3. If possible, the DBA examines the space usage for each database on the failed device. The same defaults should be set when the new database device(s) is(are) created.
4. The DBA drops the database(s) on the failed device, then the database device itself is dropped.
5. The DBA initializes a new database device. This is why it is important to keep the device creation scripts.
6. The DBA recreates each user database on the new database device. This is why it is important to keep the database creation scripts.
7. Each database is reloaded with data from the database backups and the transaction log backups. It is this procedure that is detailed below.
8. The DBA notifies the System Administrator when the database restoration is complete.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database device which has failed
- Name of replacement device
- Name of the databases which reside on the failed device
- Name of the backup volumes
- Name of the dump files on the backup volumes



Load Database and Load Transaction commands

Manual recovery of a user database is performed by the System Administrator by the use of the **LOAD DATABASE** and **LOAD TRANSACTION** commands. For issues concerning the **master** database, please consult your System Administrator's Guide for assistance. It is recommended that any user database to be recovered be dropped and created with the **for load** option

Earlier, the database creation and alteration scripts were saved. These now need to be combined into one script which will re-create the user database with the **for load** option. The **for load** option will insure the success of the **LOAD DATABASE** and **LOAD TRANSACTION** commands.

BulkCopy Utility

The **bcp** (BulkCopy) utility is located in the **\$SYBASE/bin** directory and is designed to copy data to and from SQL Server databases to operating system files.

In general, you must supply the following information for transferring data to and from SQL Server:

- Name of the database and table
- Name of the operating system file
- Direction of the transfer (in or out)

In order to use **bcp**, you must have a SQL Server account and the appropriate permissions on the database tables and operating system files that you will use. To copy data into a table, you must have **insert** permission on that table. To copy data out to an operating system file, you must have select permission on the following tables:

- The table being copied
- sysobjects
- syscolumns
- sysindexes

The script is located in **\$SYBASE/scripts** directory:

- **ins_syn_sql** is the script that copies database table to or from operating system file in a user- specified format.
- Syntax: **bcp [[database_name.]owner]table_name{in | out}**

User Database Recovery Scenario

The database **UserDB** was created using the following:

```
create database UserDB
on data_dev1 = 100
log on tx_log1 = 50
```

It can be modified using the following excerpt:

```
alter database UserDB
on data_dev1 = 50
```

For the purposes of this example, the full database backup and transaction log dumps were successful and located in \$SYBASE/sybase.dumps/dumpdir & \$SYBASE/sybase.dumps/dump_log directories.

-
- 1 At the UNIX prompt, type **cd \$SYBASE/scripts/**, then press **Return**.
 - 2 At the UNIX prompt, type **cp template.sql userdb_for_load.sql**
 - 3 Using the text editor of your choice, open the file created above and insert the following:
create database UserDB on data_dev2=100 log on tx_log2=50 for load
go
alter database UserDB on data_dev3=50
go
 - 4 When complete, save the file.
 - 5 At the UNIX prompt, type:

isql -Usa -Sservername -iuserdb_for_load.sql -ouserdb_for_load.out

then press **Return**.
 - 6 At the **Password:** prompt, type the *SybaseSAPassword*, then press **Return**.
 - Remember the *SybaseSAPassword* is case sensitive.
 - The database is recreated
 - 7 At the UNIX prompt, type **more userdb_for_load.out**
 - This allows you to view the output file for errors.
 - 8 At the UNIX prompt, type **isql -Usa**, then press **Return**.
 - 9 At the ISQL prompt, type **LOAD DATABASE UserDB from**
“\$SYBASE/sybase.dumps/dumpdir/UserDB.dat, then press **Return**; type **go**, then press **Return**.
 - This loads the database data back into the database.

10 At the ISQL prompt, type **LOAD TRANSACTION UserDB from \$SYBASE/sybase.dumps/dumpdir**, then press **Return**; type **go**, then press **Return**.

- This loads the transaction log back into the database.
-

SQL Server Login Accounts and Privileges

In order to connect to a SQL Server a login must be created by the DBA. That login must then be assigned to particular database(s) that the user will access. All login details are stored in the syslogins table in the **master** database.

Providing access to SQL servers and their databases consists of the following steps:

- A server login account for a new user is created.
- The user is added to a database and optionally assigned to a group.
- The user or group is granted permissions on specific commands and database objects.

The procedure involves creating a script that uses three SQL stored procedures:

- **sp_addlogin** - adds a new login name to the master database but does not grant access to any user database.
- **sp_adduser** - adds access capability to the defined database(s).
- **sp_helpuser** - checks the master database for information about database user logins on the system.

By creating and storing the script file, you can easily restore all database users and their assigned databases in case of catastrophic system failure.

Receive Approval For Account Creation

The SQL Server Logins and Privileges process (Figure 7) begins when the requester fills out a “SQL Server Login Account Request Form” and submits it to his or her supervisor (Figure 8).

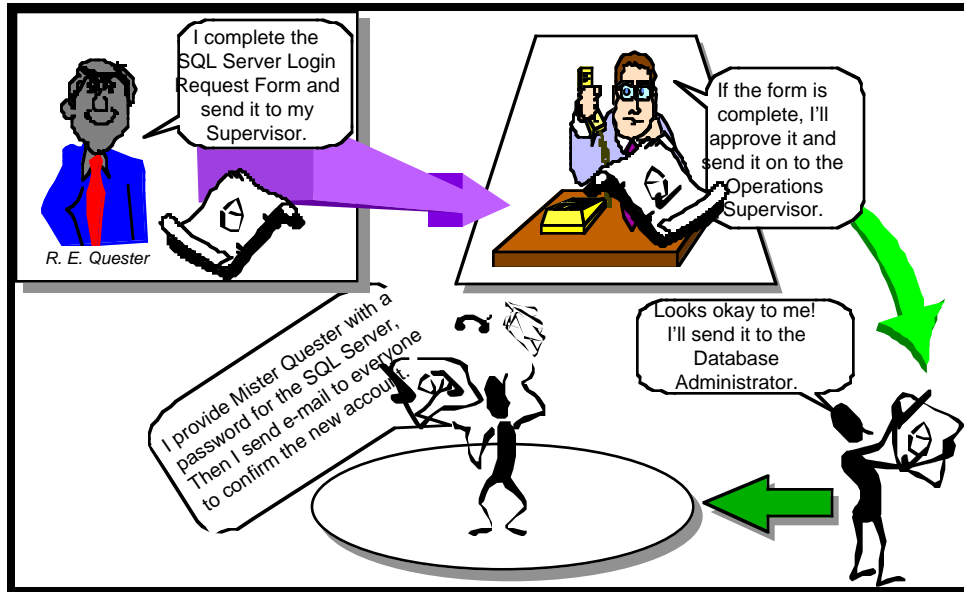


Figure 7. SQL Server login approval process.

SQL Server Login Account Request	
REQUESTER INFORMATION:	
Name: _____	Group: _____
UNIX ID: _____	Office Phone Number: _____
E-Mail Address: _____	Office Location: _____
Database(s) to be accessed: _____	
Permissions required for database objects: _____	
Justification: _____	
Date of Request: _____ Date Required: _____	
Supervisor Approval: _____	Date: _____
Ops Supervisor Approval: _____	Date: _____

Figure 8. SQL Server login account request form.

If it is complete and if the request for a new or modified SQL Server login account fits within policy guidelines, the Ops Super approves the request and forwards the request form to the DBA to implement. After the user's login account is created, the DBA provides the user with a password to use for logging onto their SQL Server login. The user can change the initial

password if local DAAC policy requires, or if he/she prefers to select his/her own password. The DBA also sends an e-mail message to the user's supervisor informing him/her that the user's SQL Server login was created.

Create SQL Server Login Accounts

The second step in creating the fully functional database user involves the DBA assigning a SQL Server logname and password for the user. The **sp_addlogin** command is used to perform this task. It will be included in the creation script as part of the procedure.

Add User to Database(s)

After the DBA determines the logname and password for the new user, the user must be added to the databases that he/she will use. The **use** and **sp_adduser** commands are used to perform this task. They will be included in the creation script as part of this procedure.

Create an SQL Server Login Procedure

- 1 Log into the server on which the new user database is to be created by typing: **telnet *ServerName*** or **rlogin *ServerName*** or **rsh *ServerName***, then press **Return**.
 - If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
 - If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.
- 2 If a **Login:** prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.
 - A password prompt is displayed.
- 3 Enter ***YourPassword***, then press **Return**.
 - Remember that ***YourPassword*** is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
- 4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber:0.0*** or **setenv DISPLAY *ServerName:0.0***, then press **Return**.
- 5 At the UNIX prompt, type **su sybase**, then press **Return**.
 - A password prompt is displayed.
- 6 Enter ***SybasePassword***, then press **Return**.
 - Remember that ***SybasePassword*** is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.

- 7 At the UNIX prompt, type **cd scripts**, then press **Return**.
- 8 At the UNIX prompt, type **cp template.sql UserName.sql**, then press **Return**.
 - This creates a new SQL script file into which you will type the appropriate commands to create the new user login.
- 9 Using the text editor of your choice, edit **UserName.sql** (Figure 9) and make changes to information enclosed in brackets (including the brackets) as appropriate:

```
/* **** */
/* name: [template.sql] */
/* purpose: */
/* written: */
/* revised: */
/* reason: */
/* **** */
sp_addlogin [login], [password]
go
use [default database]
go
sp_adduser [login]
go
sp_helpuser
go
```

Figure 9. Sample template.sql file for new database user login.

- 10 After the changes have been made, save the file according to the rules of the text editor.
- 11 At the UNIX prompt, type:
isql -Usa -SServerName -iUserName.sql -oUserName.out
then press **Return**.
 - **ServerName** is the name of the server on which the **master** database is stored.
 - **UserName.sql** is the name of the script file you created in step 9.
 - **UserName.out** is the filename of the script's output for confirmation and/or troubleshooting purposes.
 - The system will prompt you for a password.
- 12 At the **Password:** prompt, type the **SybaseSAPassword**, then press **Return**.
 - Remember the **SybaseSAPassword** is case sensitive.
- 13 When the UNIX prompt is again displayed the process is complete.
- 14 At the UNIX prompt, type **more UserName.out** , then press **Return**.

- This allows you to view the *UserName.out* file to confirm that the user login has been created or to check for user login creation errors.
-

Granting or Revoking Database Access Privileges

Permissions are used to control access within a database. The DBA uses the **grant** and **revoke** statements to accomplish this. There are two types of permissions within a database, **Object** and **Command**. In general, **Object** privileges control select, insert, update, delete, and execute permissions on tables, views, and stored procedures and **Command** permissions control access to the **CREATE** statements for databases, defaults, procedures, rules, tables, and views.

The syntax for the **grant** and **revoke** statements are quite similar:

grant *privilege(s)* **to** [public | name_list | role_name]

revoke *privilege(s)* **from** [public | name_list | role_name]

The privileges that may be granted or revoked include:

- select
- update
- insert
- delete
- references
- executeIt is recommended that the DBA store the privilege granting and revoking commands in “.sql” files in the \$SYBASE/scripts directory along with their results.

Grant or Revoke Database Access Privileges Procedure

- 1 Determine the privileges that you want to grant and to which user(s).
- 2 Log into the server on which the new user database is to be created by typing: **telnet** *ServerName* or **rlogin** *ServerName* or **rsh** *ServerName*, then press **Return**.
 - If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
 - If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.
- 3 If a **Login:** prompt appears, log in as yourself by typing: *YourUserID*, then press **Return**.
 - A password prompt is displayed.
- 4 Enter *YourPassword*, then press **Return**.

- Remember that *YourPassword* is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
- 5 Set display to current terminal by typing: **setenv DISPLAY IPNumber:0.0** or **setenv DISPLAY ServerName:0.0**, then press **Return**.
- 6 At the UNIX prompt, type **su sybase**, then press **Return**.
- A password prompt is displayed.
- 7 Enter *SybasePassword*, then press **Return**.
- Remember that *SybasePassword* is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.
- 8 At the UNIX prompt, type **isql -Usa**, then press **Return**.
- 9 At the **Password:** prompt, type the *SybaseSAPassword*, then press **Return**.
- Remember the *SybaseSAPassword* is case sensitive.
- 10 If you are granting privileges, at the ISQL prompt, type **grant Privilege to Logname**, then press **Return**.
- If more than one privilege is to be granted, repeat this step on subsequent lines.
- 11 If you are revoking privileges, at the ISQL prompt, type **revoke Privilege from Logname**, then press **Return**.
- If more than one privilege is to be revoked, repeat this step on subsequent lines.
- 12 When all privileges have been assigned, type **go**, then press **Return**.
-

Database Tuning and Performance Monitoring

Day-to-day operation of a complex database system requires the DBA to actively and continuously monitor the performance of the database servers for degradation of processing speed, available disk space, and connectivity, to name just a few of the many configurable parameters. This section discusses some of the SQL tools that are available to assist the DBA in performing system tuning and performance monitoring.

Configure SQL Server

System 10 SQL Server has about 40 configurable parameters that can be set permanently or for single runs of a program. Using the **sp_configure** command, you can display a list of the names and current values of the parameters (Figure 10). Please refer to the Sybase System Administrator's guide (URL = <http://sybooks.sybase.com/cgi-bin/nph-dynaweb/srv11007/sysadm/1.toc>) for specific information about what each of the parameters controls.

The column titled **name** is the description of the variable. The column titled **minimum** is the minimum allowable configuration setting. The column titled **maximum** is the theoretical maximum value to which the configuration option can be set. The actual maximum value is dependent on the specific platform and available resources to the SQL Server. The column titled **config_value** reflects the current system default values. The column titled **run_value** reflects the values the system is currently utilizing, which may be different from the default values.

name	minimum	maximum	config_value	run_value
recovery interval	1	32767	0	5
allow updates	0	1	0	0
user connections	5	2147483647	0	25
memory	3850	2147483647	0	5120
open databases	5	2147483647	0	12
locks	5000	2147483647	0	5000
open objects	100	2147483647	0	500
procedure cache	1	99	0	20
fill factor	0	100	0	0
time slice	50	1000	0	100
database size	2	10000	0	2
tape retention	0	365	0	0
recovery flags	0	1	0	0
nested triggers	0	1	1	1
devices	4	256	0	10
remote access	0	1	1	1
remote logins	0	2147483647	0	20
remote sites	0	2147483647	0	10
remote connections	0	2147483647	0	20
pre-read packets	0	2147483647	0	3
upgrade version	0	2147483647	1002	1002
default sortorder id	0	255	50	50
default language	0	2147483647	0	0
language in cache	3	100	3	3
max online engines	1	32	1	1
min online engines	1	32	1	1
engine adjust interval	1	32	0	0
cpu flush	1	2147483647	200	200
i/o flush	1	2147483647	1000	1000
default character set id	0	255	1	1
stack size	20480	2147483647	0	28672
password expiration interval	0	32767	0	0
audit queue size	1	65535	100	100
additional netmem	0	2147483647	0	0
default network packet size	512	524288	0	512
maximum network packet size	512	524288	0	512
extent i/o buffers	0	2147483647	0	0
identity burning set factor	1	9999999	5000	5000
allow sendmsg	0	1	0	0
sendmsg starting port number	0	65535	0	0

Figure 10. sp_configure sample output

In order to perform the procedure, the DBA must have obtained the following information:

- Name of server to be configured
- New values for configuration variables.

To set SQL Server configuration variables, the DBA must have **sa_role** (SQL Server) permissions. To set SQL Server configuration variables **allow updates**, **audit queue size**, **password expiration interval**, or **remote access**, **sso_role** (SQL Server) is also required.

Some parameter values take affect as soon as you reset the value. Others, which involve memory reallocation, do not change until you reset the value and then reboot SQL Server.

Configure SQL Server Procedure

- 1 Log into the server which will be reconfigured by typing: **telnet *ServerName*** or **rlogin *ServerName***, then press **Return**.
 - 2 If a **Login:** prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.
 - A password prompt is displayed.
 - 3 Enter ***YourPassword***, then press **Return**.
 - Remember that ***YourPassword*** is case sensitive.
 - You are authenticated as yourself and returned to the UNIX prompt.
 - 4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber:0.0*** or **setenv DISPLAY *ServerName:0.0***, then press **Return**.
 - 5 Begin the SQL session by typing at the UNIX prompt **isql -S*ServerName***, then press **Return**.
 - 6 Type **sp_configure**, press **Return**, type **go**, then press **Return**.
 - A list of the configurable parameters, their maximum and minimum values, the current setting and the default values is displayed.
 - 7 After determining which parameter(s) to reset, type:

sp_configure "*ParameterName*", *NewValue*

then press **Return**; type **go**, then press **Return**.
 - ***ParameterName*** is the name from the list displayed in step 6 above. Be sure to enclose the name in double quotes.
 - ***NewValue*** is the numeric value that you want to assign to ***ParameterName***.
 - 8 Repeat step 7 above for each parameter you want to reconfigure.
 - 9 When complete, type **quit** at the ISQL prompt, then press **Return**.
 - You are returned to the UNIX prompt.
-

Memory Utilization and Configuration

Monitoring and configuring memory is probably the most important of the configurable parameters. If more data can be accessed and worked on in the memory cache, then the system will perform faster. However, configuring a large amount of the cache to an underutilized procedure can lead to a degradation in system performance.

When SQL server starts up, it refers to the **memory** system configuration (Figure 11) to determine the number of pages of real memory that the SQL server requests from the operating system when it comes up.

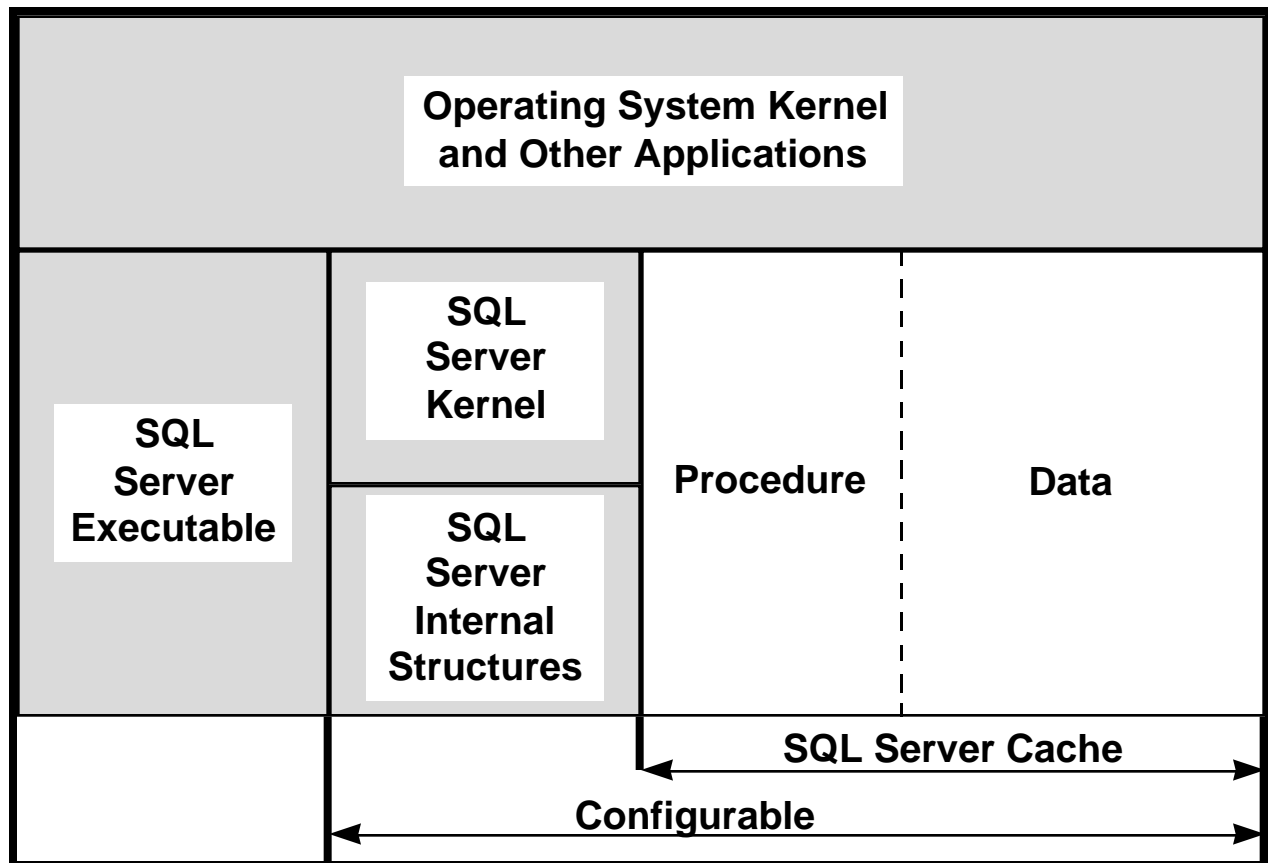


Figure 11. Physical memory allocation

An SQL Server is configured when the Database Administrator determines that a customization or fine tuning is required to optimize memory allocation or performance.

AUTOSYS (Unbundled Sybase):

Implementation: Dual Event Server with Shadow Event Processor.

In VATC, the Primary Event Server/Processor is on t1pls02 and the Shadow (Secondary) Event Server/Processor is on t1pls01. This type of implementation requires a “Third Machine” which is also a AutoSys Remote Agent (a.k.a. AutoSys Client). The “Third Machine” is t1pls02. The “Third Machine” is used to resolve contentions. The AutoSys Instance name and the AutoSys database name is “VAT”.

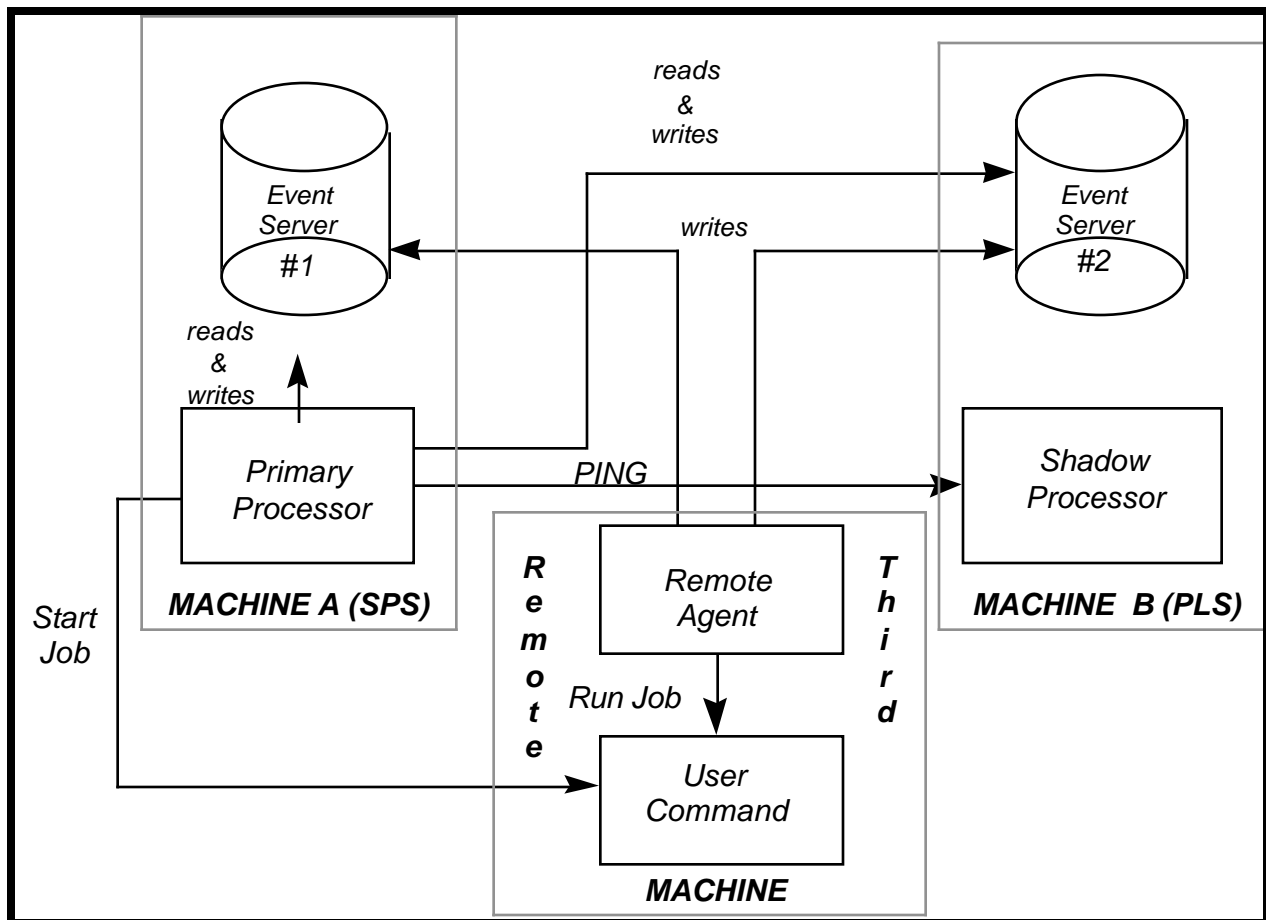


Figure 12. Topology for Running Two Event Processors

In case the Primary Event Processor fails, the Secondary Event Processor takes over event processing. The Secondary Event Processor is usually sitting idle and only listening to ping messages from the Primary Event Processor. When the Secondary Event Processor doesn't receive any message from the Primary Event Processor, it assumes that the Primary Event Processor is down. It then checks with the "Third Machine" defined in `config.instance-name` file to make sure that the Primary Event Server is actually down, and takes control of event processing.

All three machines will have the same `config.instance-name` file. A sample of entries in `config.instance-name` is as follows:

Sample config.*instance-name* file:

```
#####
#
#   Configuration file for AutoSys Version 3.3
#
#####
#   Modified : Clarence Pinto (8/19/97)
#           t1sps02_srvr  is Primary Autosys Server
#           t1pls01_srvr  is Secondary Autosys Server
#           Uncommented the second EventServer entry for t1pls01_srvr
#           Uncommented ThirdMachine and entered the 3rd machine name
#           Commented the first DBEventReconnet and
#           Uncommented the second DBEventReconnect
#
#####

#
#   Specify the Database(s)
#
# Sybase database example
EventServer=t1sps02_srvr:VAT
EventServer=t1pls01_srvr:VAT
.
.
.
.
#
# Number of times for Event Processor
#   to attempt to Reconnect to Event Servers
#
#DBEventReconnect=50
#
# USE the following for Dual DataBases
DBEventReconnect=50,5
.
.
```

```

.

#
# Machines for chk_auto_up to inspect to see if an
# Event Processor is running there.
#
#EDMachines=host1,host2,host3
EDMachines=t1sps02,t1pls01,t1pls02,t1spg01,t1ais01

#
# For Shadow Event Processor, the Third Machine to resolve
# contention problems between Event Processors
ThirdMachine=t1pls02

.
.
.
.
.
#####

```

You only start the AutoSys Event Processor on the Primary machine with the following options:

In c shell:

- 1** % cd /usr/ecs/OPS/COTS/autotreeb/autouser then press **Return**.
- 2** % setenv autosys.csh.*primary-machine-name* then press **Return**.
- 3** % eventor -M *secondary-machine-name* then press **Return**.

In bourne shell:

- 1** cd /usr/ecs/OPS/COTS/autotreeb/autouser then press **Return**.
 - 2** # autosys.sh.*primary-machine-name* then press **Return**.
 - 3** # eventor -M *secondary-machine-name* then press **Return**.
-

By doing the above command, AutoSys starts the Primary Event Processor on t1sps02 (Primary machine) and automatically starts the Shadow Event Processor on t1pls01 (Secondary machine). To check if AutoSys is working the way it should, do the following:

In C shell:

- 1 % cd /usr/ecs/OPS/COTS/autotreeb/autouser then press **Return**.
- 2 % setenv autosys.csh.*primary-machine-name* then press **Return**.
- 3 % chk_auto_up then press **Return**.

In bourne shell:

- 1 # cd /usr/ecs/OPS/COTS/autotreeb/autouser then press **Return**.
 - 2 # autosys.sh.*primary-machine-name* then press **Return**.
 - 3 # chk_auto_up then press **Return**.
-

The results should like this:

```
Attempting (1) to Connect with Database: t1sps02_srvr:VAT
```

```
*** Have Connected successfully with Database: t1sps02_srvr:VAT. ***
```

```
Connected with Event Server: t1sps02_srvr:VAT
```

```
Attempting (1) to Connect with Database: t1pls01_srvr:VAT
```

```
*** Have Connected successfully with Database: t1pls01_srvr:VAT. ***
```

```
Connected with Event Server: t1pls01_srvr:VAT
```

Checking Machine: t1sps02

Primary Event Processor is RUNNING on machine: t1sps02

Checking Machine: t1pls01

Shadow Event Processor is RUNNING on machine: t1pls01

Checking Machine: t1pls02

No Event Processor is RUNNING on machine: t1pls02

Checking Machine: t1spg01

No Event Processor is RUNNING on machine: t1spg01

Checking Machine: t1ais01

No Event Processor is RUNNING on machine: t1ais01

This page intentionally left blank.

Database Security and Auditing

Database Security and Auditing

The following statements represent examples for performing security and auditing:

-
- 1 Run sybinit and install auditing.
 - 2 Add a login for auditing:
sp_addlogin ssa, ssa_password, sybsecurity
use sybsecurity
sp_changedbowner ssa
sp_role "grant", sso_role, ssa
 - 3 Enable auditing:
sp_auditoption "enable auditing", "on"
sp_auditlogin loginname, "cmdtext", "on"
 - 4 To Test:
create a table in a database with one field
grant all on the table for the loginname
log into isql using the loginname
insert a record into the table
log into isql as ssa
select * from sysaudits where loginname =
"loginname"
-

This page intentionally left blank.

Integrity Monitoring

Integrity Monitoring

The Database Consistency Checker (**dbcc**) is a set of utility commands for checking the logical and physical consistency of a database. Use the **dbcc** commands:

- As part of regular database maintenance (periodic checks run by the System Administrator). These checks can detect, and often correct, errors before they affect a user's ability to use SQL Server.
- To determine the extent of possible damage after a system error has occurred.
- Before backing up a database.
- Because you suspect that a database is damaged. For example, if using a particular table generates the message "Table corrupt", you can use dbcc to determine if other tables in the database are also damaged.

The integrity of the internal structures of a database depends upon the System Administrator or Database Owner running database consistency checks on a regular basis. Two major functions of dbcc are:

- Checking allocation structures (the commands checkalloc, tablealloc, and indexalloc.
- Checking page linkage and data pointers at both the page level and row level (checktable and checkdb). The next section explains page and object allocation and page linkage.

dbcc is used in the database backup scripts to determine if the database is properly configured and without errors. If errors occur, the backup will not proceed and a message is sent to the DBA informing him/her of the problem.

This page intentionally left blank.

Troubleshooting

Diagnosing Database System Problems

Symptoms of a Damaged master Database

A damaged master database can be caused by a media failure in the area on which master is stored, or by internal corruption in the database. A damaged master database makes itself known in one or more of these ways:

- SQL Server cannot start.
- There are frequent or debilitating segmentation faults or input/output errors.
- **dbcc** (the database consistency checker) reports damage during a regularly-scheduled check of your databases.

This page intentionally left blank.

Practical Exercises

Introduction

These practical exercises are presented in “day-in-the-life” scenarios relating to database administration activities. They represent real situations that you, as database administrator, are likely to encounter on a day-to-day basis.

Equipment and Materials

A functioning Version 2.0 system.

Starting and Stopping SQL Server

1. Check to see that the SQL Server processes are running. If they are not running, start them. If they are running, shut them down and restart them.

Database Devices

1. Create the script file(s) that will create a database device on the server to be announced from the podium. The following parameters should be met:
 - Device Name = class_device
 - Physical Name = TBA
 - Device Size = 100 MB (be sure to convert this to blocks!)
 - Virtual Device Number = you must determine this number.
2. Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.
3. If time and space permit (to be determined by the instructor), perform the creation of this device.

User Databases

Margaret Janis, a data scientist, has requested that a small database be created so she can track 50 new experiments.

1. Create the script file(s) that will create a user database on the server to be announced from the podium. The following parameters should be met:
 - Database Name = mjanisdb
 - Database Device Name = class_device
 - Database Size = 50 MB
 - Transaction Log Size = 5 MB
 - Transaction Log Device = default_device
2. Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.
3. If time and space permit (to be determined by the instructor), perform the creation of this database.

Backup and Recovery

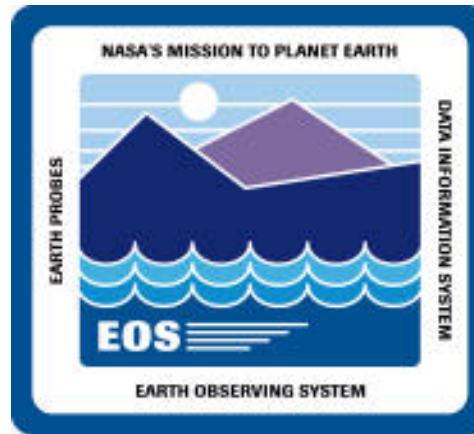
1. Perform a manual backup of the autosys database. Send the backup to the autosys_backup server.
2. Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.
3. Assume that the database device you created at the beginning of these exercises has failed. Perform all the steps required to recreate the database device and restore the database to full functionality.

Slide Presentation

Slide Presentation Description

The following slide presentation represents the slides used by the instructor during the conduct of this lesson.

This page intentionally left blank.



DATABASE ADMINISTRATION

ECS Version 2.0 Training

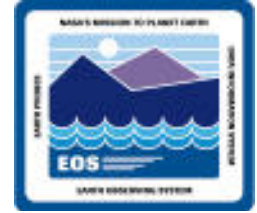
625-CD-011-001

Objectives

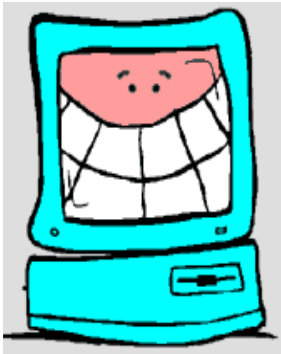


- **Create new database devices**
- **Allocate disk space**
- **Maintain database segments**
- **Maintain transaction logs, error logs**
- **Maintain the interfaces file (Configure SQL Server)**
- **Startup and shutdown SQL servers**
- **Startup and shutdown Back Server**
- **Startup and shutdown Monitor Server**
- **Security and monitoring**
- **Product Installation and Disk Storage Management**
- **Backup and recovery**
- **Configuring, tuning, and monitoring**

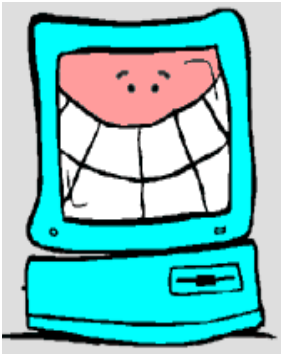
DBA Tasks and Functions



- Perform database backup, transaction log maintenance, and database recovery.
- Monitor and tune the physical allocation of database resources.
- Maintain user accounts.
- Create user registration and account access control permissions in the security database.
- Work with data specialists on DB design, data sets, and metadata management.

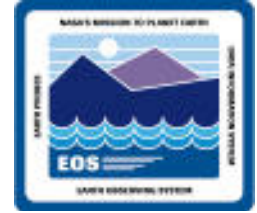


Sybase Directory Structure

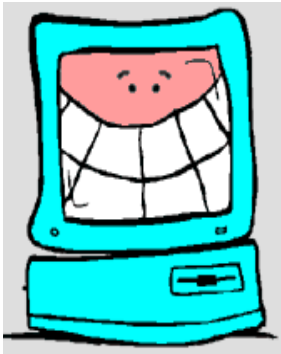


Directory	Contains
\$SYBASE/bin	Utilities necessary to load, run, and access the server
\$SYBASE /install	Files used to start dataservers, backupserver and to record server messages (errorlogs)
\$SYBASE /scripts	Root directory for all script files executed on the server
\$SYBASE/scripts/mk_DB_database.sql (e.g. database = SubServer)	SQL script files used to create databases on the server
\$SYBASE/scripts/add_devices.sql	SQL script file (disk init) used to map physical storage to a logical name.
\$SYBASE/scripts/alter_DB_tempdb.sql	SQL script file used to alter the database.
\$SYBASE /scripts/del_user	Script file used to delete logins and remove them from the users database.
backup directory: \$SYBASE/sybase.dumps	Root directory that contains all backup subdirectories. It is recommended, but not required, that this directory be on a separate physical disk

Naming Conventions



- The file name should indicate the function and/or content of the object regardless of the length of the file name.
- Only easily understandable abbreviations should be used.



Parts of names are separated by the underscore character(_).

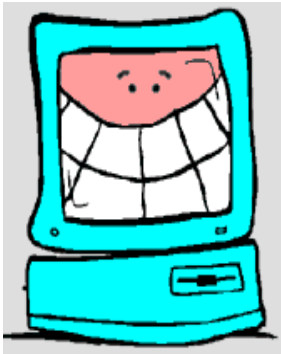
Only one optional suffix is permitted and is appended to the file name by a period (.).

- The full path of the object is considered to be part of the name.

Version 2.0 Databases



Version 2.0 databases are divided into two categories:



Production Databases

System Management Databases

B.0 Databases



ICLHW
Ingest

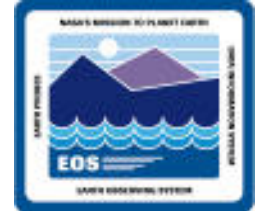
DMGHW
Advertising

ACMHW
*Storage Management Pull Monitor
Metadata*

PLNHW
Planning and Data Processing Subsystem

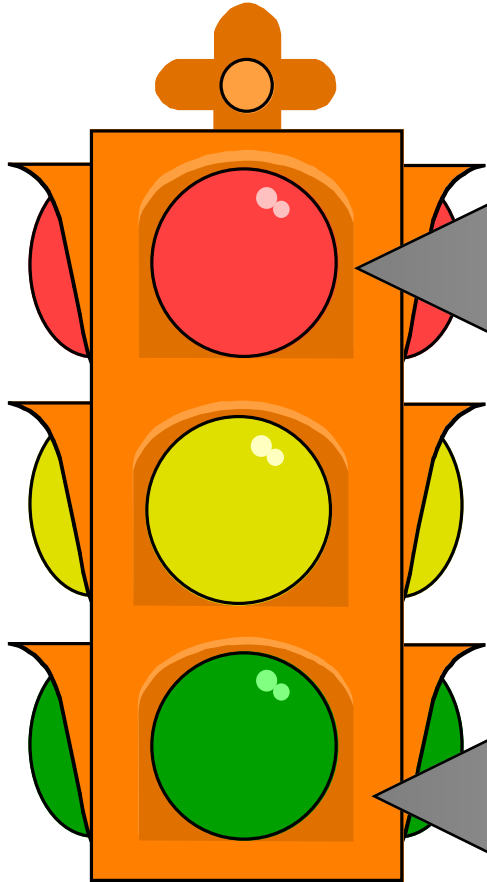
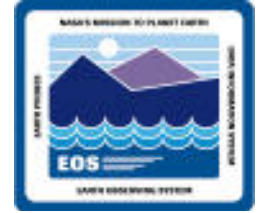
ASTER LUT
*Aster Lookup Table
(EDC only)*

Maintaining the Interfaces file



- ***Adding an entry to the interfaces file
(When a new SQL Server is created)***
- ***Modifying an existing entry in the interface
file (When the machine that has the SQL
Server is running gets moved)***

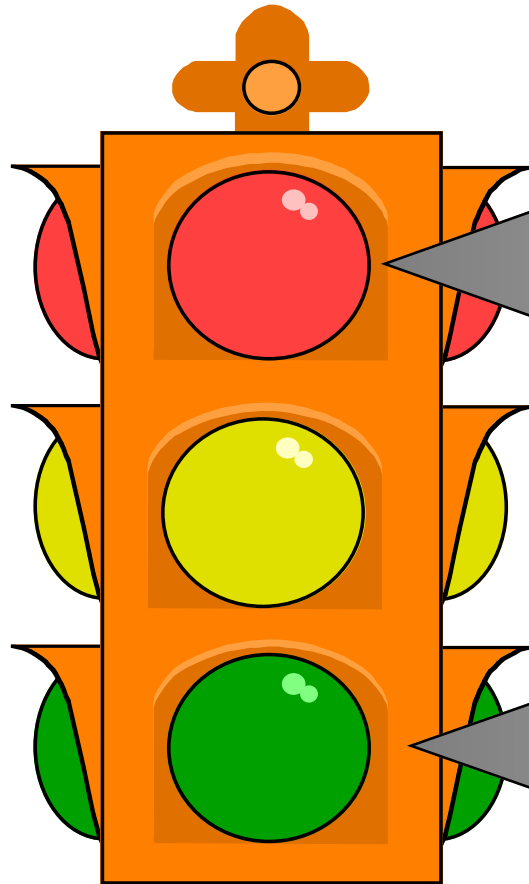
Start & Stop SQL Server



**STOP! BACKUP and
MONITOR Server must be
STOPPED before
STOPPING SQL Server**

**START the SQL Server
after installation, system
outage
or maintenance**

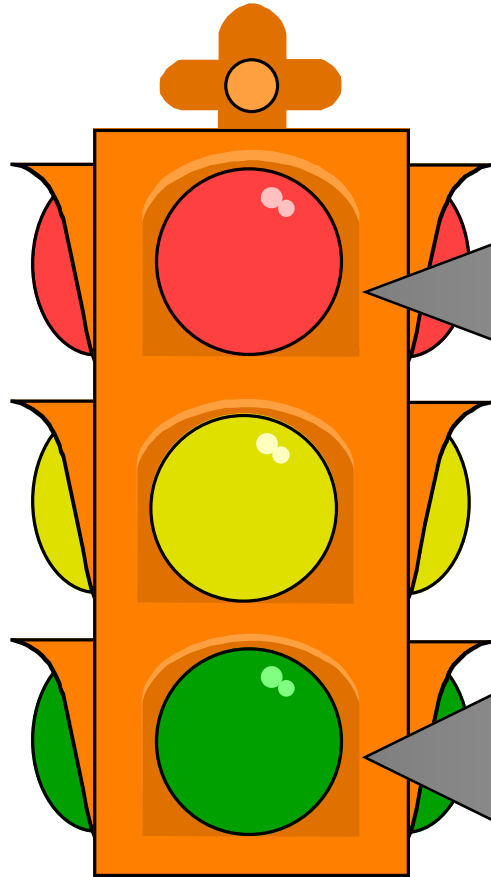
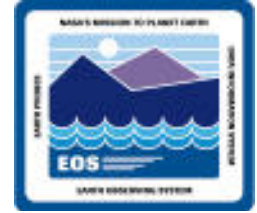
Start & Stop SQL Backup Server



STOP the **BACKUP** Server
before **STOPPING** **SQL**
Server

SQL Server must be **Up**
and **Running** in order to
START the **BACKUP** Server

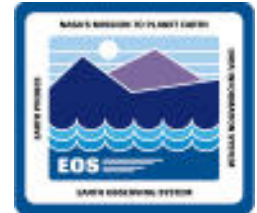
Start & Stop SQL Monitor Server



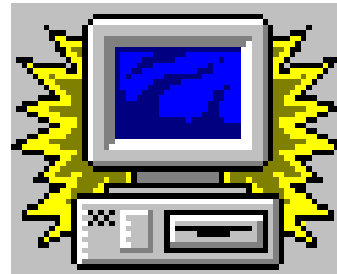
STOP the monitor server
before stopping **SQL
Server**

SQL Server must
be up and running in
order to **START** the
monitor server

Database Device



- Stores objects that make up databases
- May be:
 - a distinct physical device
 - a disk partition
 - a file
- Must be initialized first



Initializing a Database Device Command Script Template



```
/* **** */
/* name: [add_devices.sql] */
/* purpose: */
/* written: */
/* revised: */
/* reason: */
/* **** */
disk init name = [device name],
physname = "/dev/[device name]",
vdevno = [#],
size = [size]
```

Initializing a Database Device Completed Sample Script



```
/* **** */
/* name: subserver.sql */
/* purpose: tracking training program */
/* written: */
/* **** */
disk init name = subserver_data,
physname = "/dev/vx/rdisk/rootdg/SubServer_Data",
vdevno = 3,
size = 1024000
```

Create New Database Command Script Template



```
/* **** */
/* name: [mk_DB_database-name.sql]
/* purpose: * /
/* written: * /
/* revised: * /
/* reason: * /
/* **** */
create database [DBName]
on [DBDeviceName] = [DBSize_in_MB]
log on [LogDBDeviceName] = [LogSize_in_MB]
```

Create New Database Command Sample Script



```
/* **** */
/* name: mk_DB_SubServer.sql */
/* purpose: track specific data items */
/* written: */
/* **** */
create database SubServer
  on subserver_data = 100, subserver_index=50
  log on subserver_log = 50
```

User Database Request Form



User Database Request Form

REQUESTER INFORMATION:

Name: _____

Office Phone Number: _____

E-Mail Address: _____ Office Location: _____

DATABASE(S) TO BE CREATED:

JUSTIFICATION: _____

Date of Request: _____ Date Required: _____

Supervisor Approval: _____ Date: _____

Ops Supervisor Approval: _____ Date: _____

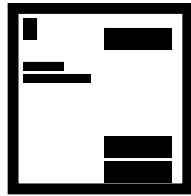
Renaming Database Command Sample



```
/* Rename database Old-database-name to New-database-name */

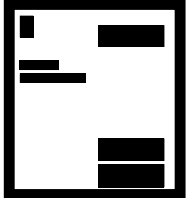
sp_dboption Old-database-name, "single user", true
go
use Old-database-name
go
checkpoint
go
use master
go
sp_renamedb Old-database-name, New-database-name
go
sp_dboption New-database-name, "single user", false
go
use New-database-name
go
checkpoint
go
use master
go
```

Servers Name:(Sybase & SQS)



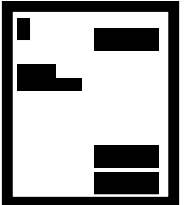
- *tlins01_svr*
- *tlins01_mon_svr*
- *tlins01_syb_backup*

T1INS01



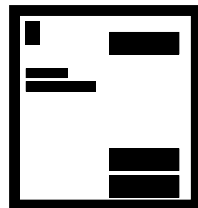
- *tlmsh01_svr*
- *tlmsh01_mon_svr*
- *tlmsh01_syb_backup*

T1MSH01



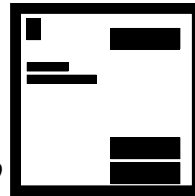
- *tlmss06_svr*
- *tlmss06_mon_svr*
- *tlmss06_syb_backup*

T1MSS06



T1SPS02

- *tlsp02_svr*
- *tlsp02_mon_svr*
- *tlsp02_syb_backup*



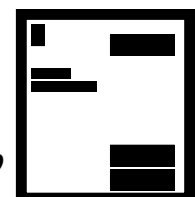
- *tlins02_svr*
- *tlins02_mon_svr*
- *tlins02_syb_backup*

T1INS02



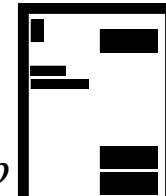
- *tlpls01_svr*
- *tlpls01_mon_svr*
- *tlpls01_syb_backup*

T1PLS01



- *tlmss06_svr*
- *tlmss06_mon_svr*
- *tlmss06_syb_backup*

T1MSS07



- *tlacg01_svr*
- *tlacg01_sqs222_svr*
- *tlacg01_mon_svr*

T1ACG01



- *tlcig01_svr*
- *tlcig01_sqs222_svr*
- *tlcig01_mon_svr*

T1ICG01

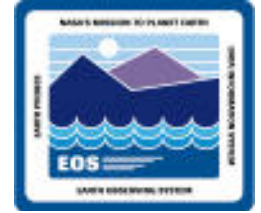


- *tlcig03_svr*
- *tlcig03_sqs222_svr*
- *tlcig03_mon_svr*

T1ICG03

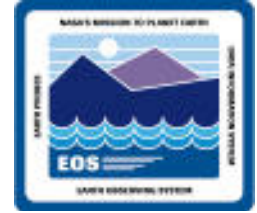
- *tlcig03_syb_backup*

Changing Password Command Sample



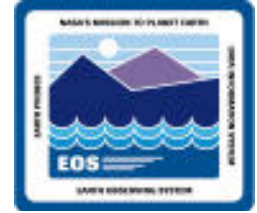
`sp_passwod old-password, new-password, user-name`

Database Segments



- **Collection of database devices or fragments available to a particular database**
- **Can have tables and indexes assigned to it**
- **Can span a set of physical devices**
- **Created when the database is created *or* when DBA deems necessary *or* as part of the database recovery procedure**

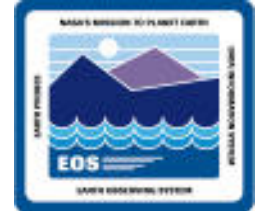
Why Database Segments?



- Reduces read/write access time
- Increases SQL Server performance
- Added administrative control over placement, size, and space usage of specific database objects



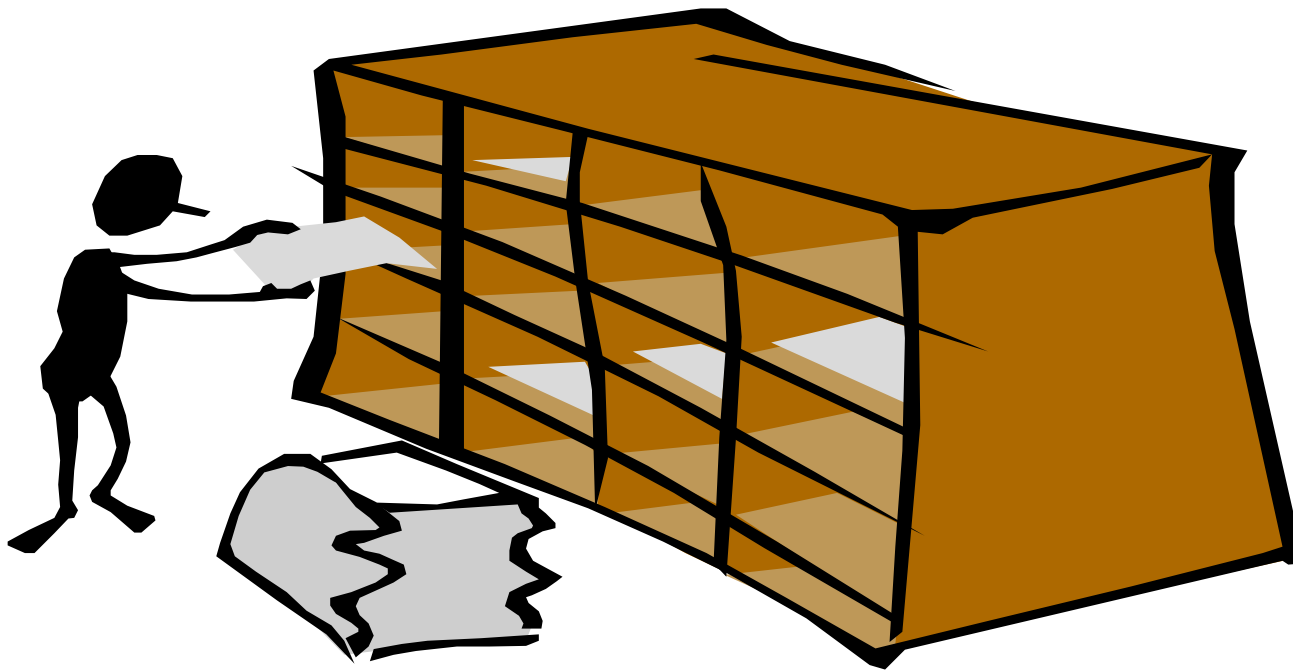
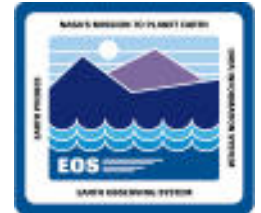
Database Segments Template File



```
/* **** */
/* name: [segment.sql] */
/* purpose: */
/* written: */
/* revised: */
/* reason: */
/* **** */
```

```
sp_addsegment [seg_name], [DBname],
[DBDevice]
go
```


Backup and Recovery

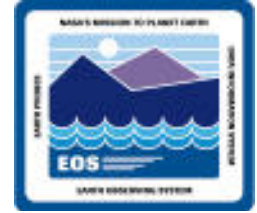


Database Data Backup



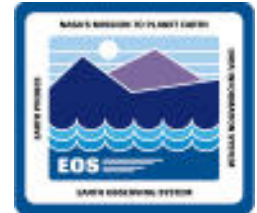
- **Databases data are backed daily**
- **Can be requested at any time**
- **Need to know the following:**
 - **Name of DB to be backed up**
 - **Name of the server on which the DB resides**
 - **Name of the backup volume**
 - **Name of the dump file on the backup volume**
- **Run daily by a UNIX Cron Job**

Database Recovery/ Database Device Restoration



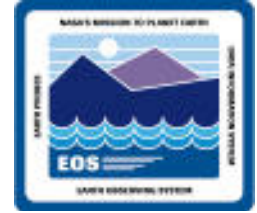
- **Device Failure verified by SA**
- **SA requests a restoration from the DBA**
- **Transaction log for each DB on the failed device is backed up**
- **DBA examines space usage of each DB on failed device.**
- **DB(s) on the failed device are deleted; then device is deleted**
- **DBA initializes new DB device**
- **DBA recreates each user DB on the new device**
- **Each DB is restored from DB backups and transaction logs**
- **DBA notifies SA when restoration is complete.**

dbcc memusage Sample Output



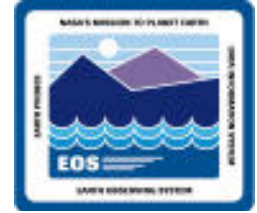
	Meg.	2K Blks	Bytes
Configured Memory:	400.0000	204800	419430400
Code size:	3.4259	1755	3592296
Kernel Structures:	5.9769	3061	6267212
Server Structures:	13.9494	7143	14627040
Cache Memory:	357.0625	182816	374407168
Proc Buffers:	0.6974	358	731272
Proc Headers:	18.8848	9669	19802112

Transaction Log Backup



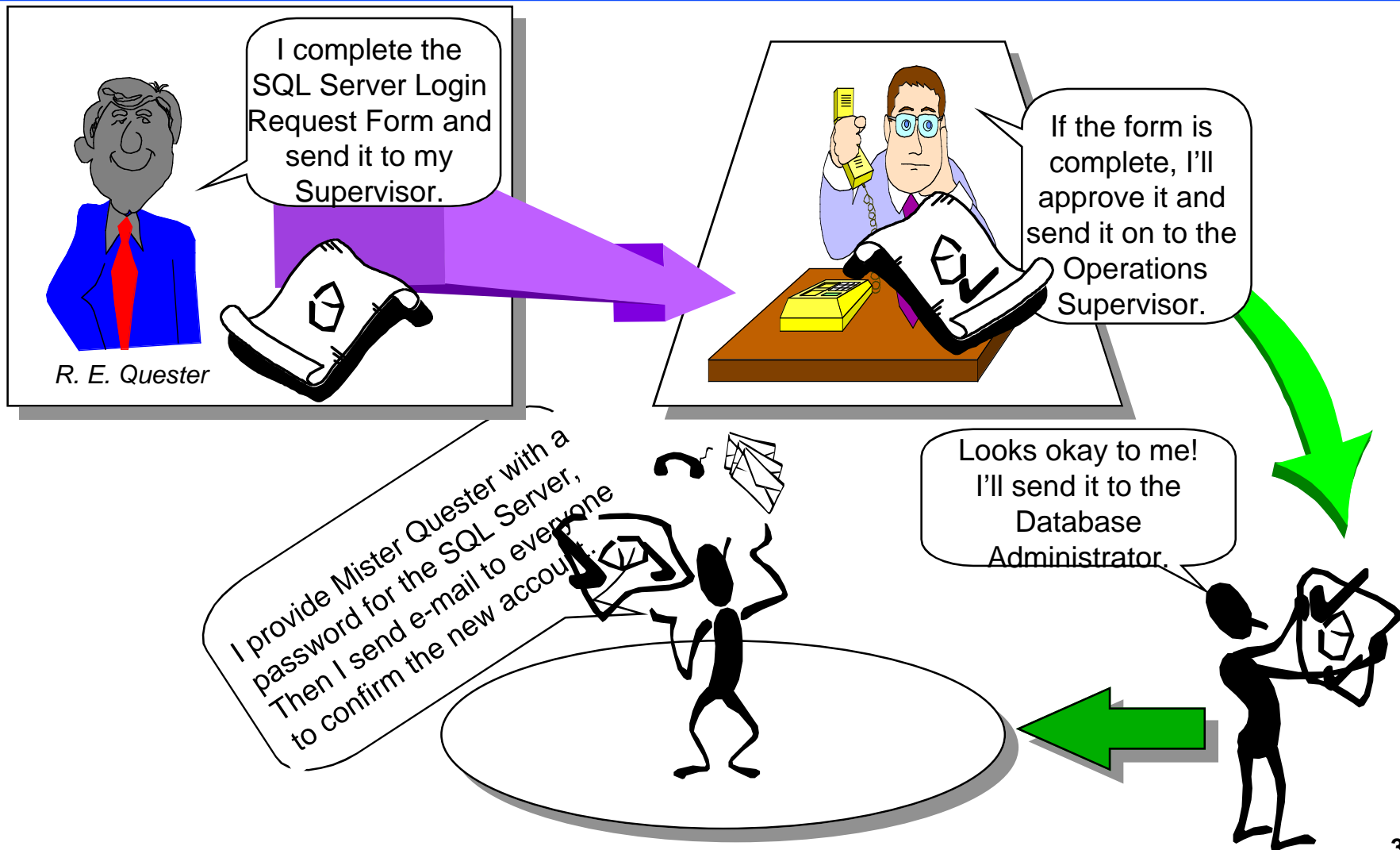
- **Transaction Log - an expanding file that records all database transactions.**
- **Used for complete recovery of DB if media fails.**
- **Maintained on a different device than DB**
- **Backed up with regular system backups, but non-scheduled backups can be performed with permission.**

Configure SQL Server



- **Customization**
- **Fine Tuning**
- **Optimize memory allocation or performance**
- **Configuration Variables:**
 - allow/deny updates
 - audit queue size
 - password expiration interval
 - remote access
- **Some values take effect immediately, others require a server reboot.**
 - When in doubt, reboot!

SQL Server Login Approval Process



SQL Server Login Account Request Form



SQL Server Login Account Request

REQUESTER INFORMATION:

Name: _____

UNIX ID: _____ Group: _____

Office Phone Number: _____

E-Mail Address: _____ Office Location: _____

Database(s) to be accessed: _____

Permissions required for database objects: _____

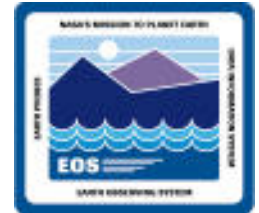
Justification: _____

Date of Request: _____ Date Required: _____

Supervisor Approval: _____ Date: _____

Ops Supervisor Approval: _____ Date: _____

Database Access Privileges



Assign a user to a group that has specific access privileges.



Assign a user object permissions.



Assign a user command permissions.

Database Tuning and Performance Monitoring



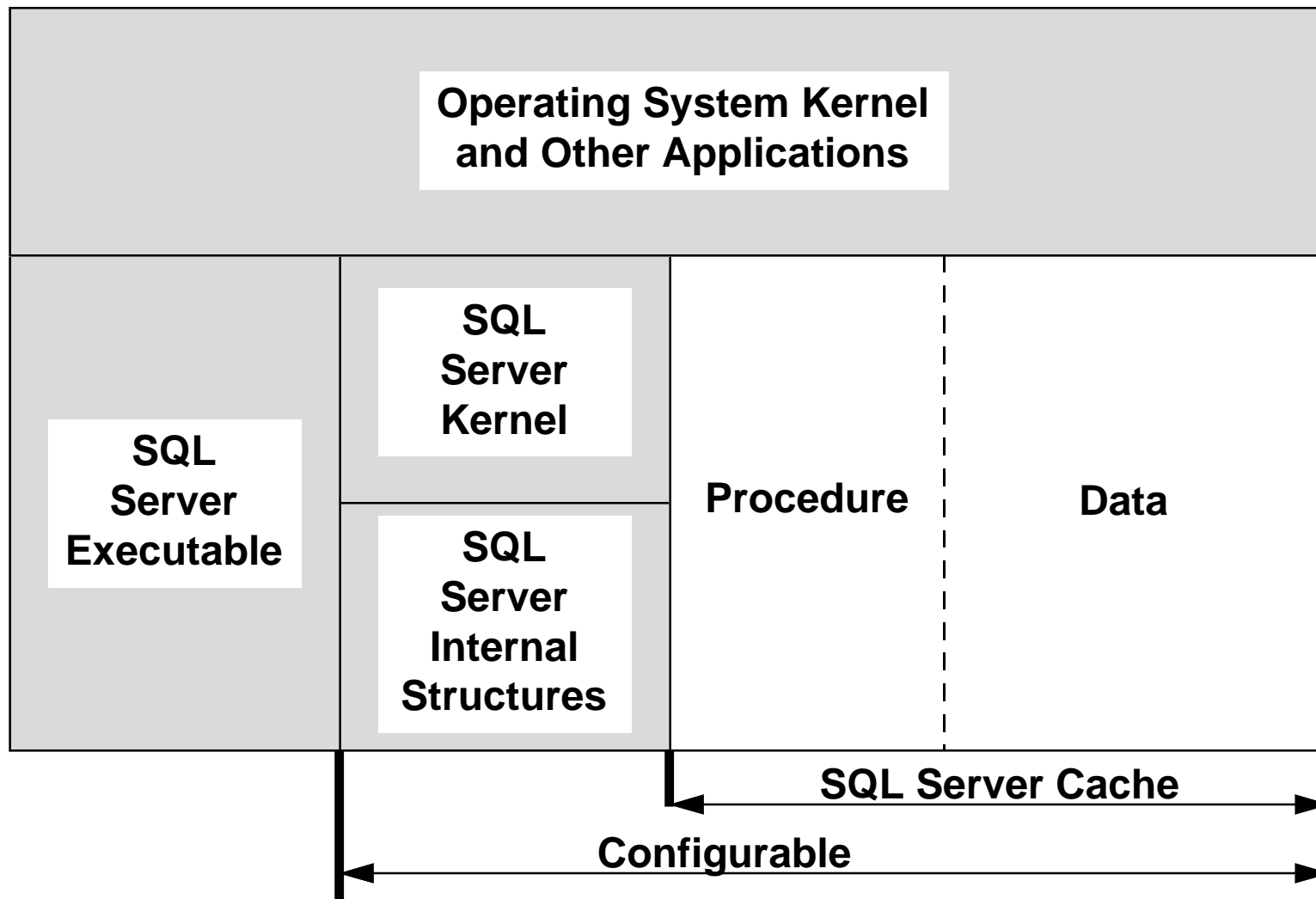
- Use *sp_config* to determine current configuration parameters and set future run values.
- Use *dbcc memusage* to determine current memory usage.
- Use *sp_spaceused* to determine how much space has been used on the device.
- Running two event processors

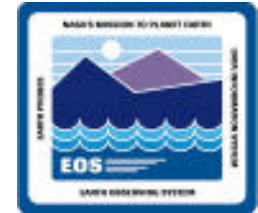
sp_config Sample Output



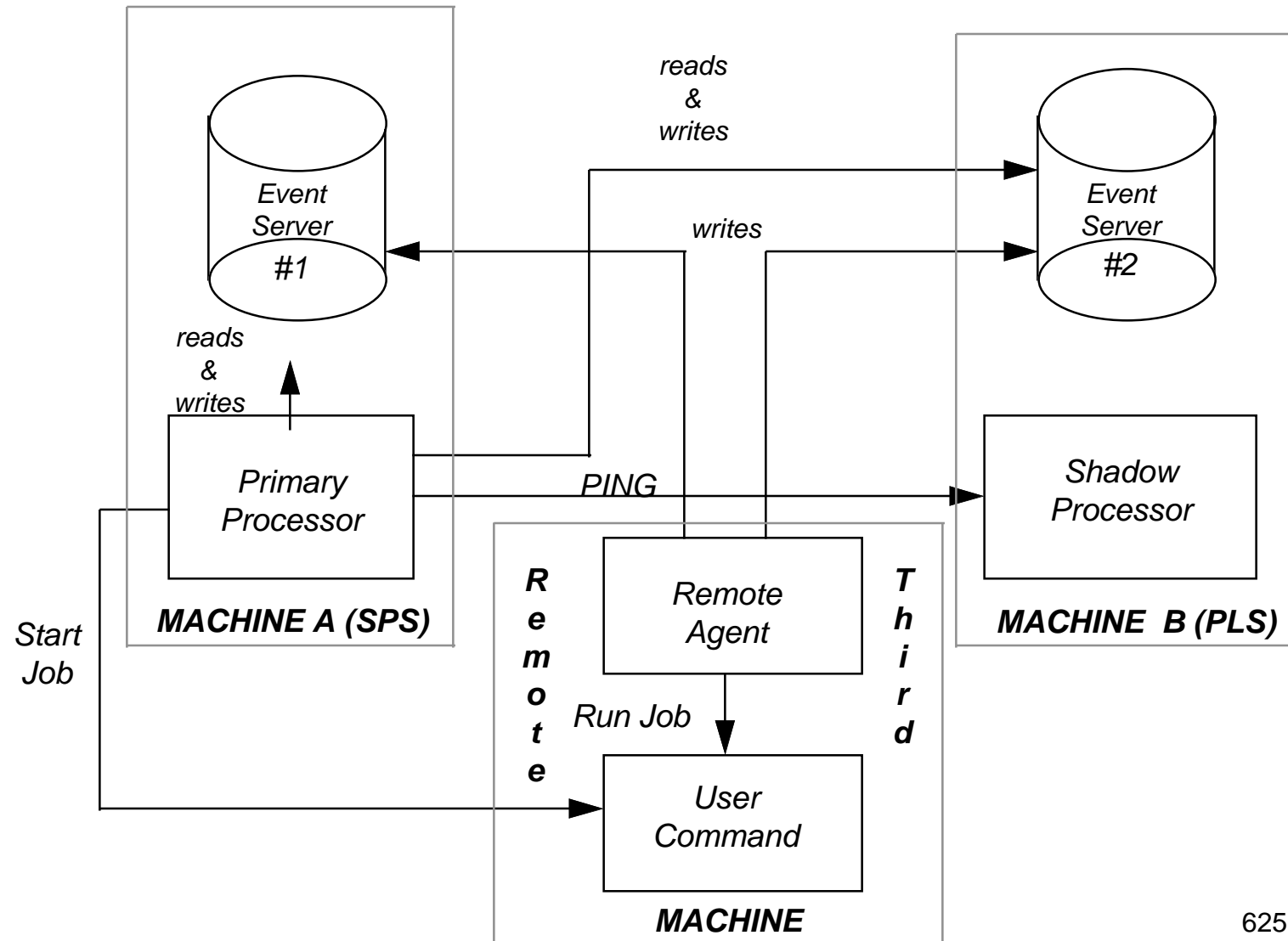
name	minimum	maximum	config_value	run_value
recovery interval	1	32767	0	5
allow updates	0	1	0	0
user connections	5	2147483647	0	25
memory	3850	2147483647	0	5120
open databases	5	2147483647	0	12
locks	5000	2147483647	0	5000
open objects	100	2147483647	0	500
procedure cache	1	99	0	20
fill factor	0	100	0	0
time slice	50	1000	0	100
database size	2	10000	0	2
tape retention	0	365	0	0
recovery flags	0	1	0	0
nested triggers	0	1	1	1
devices	4	256	0	10
remote access	0	1	1	1
remote logins	0	2147483647	0	20
remote sites	0	2147483647	0	10
remote connections	0	2147483647	0	20
pre-read packets	0	2147483647	0	3
upgrade version	0	2147483647	1002	1002
default sortorder id	0	255	50	50
default language	0	2147483647	0	0
language in cache	3	100	3	3
max online engines	1	32	1	1
min online engines	1	32	1	1
engine adjust interval	1	32	0	0
cpu flush	1	2147483647	200	200
i/o flush	1	2147483647	1000	1000
default character set id	0	255	1	1
stack size	20480	2147483647	0	28672
password expiration interval	0	32767	0	0
audit queue size	1	65535	100	100
additional netmem	0	2147483647	0	0
default network packet size	512	524288	0	512
maximum network packet size	512	524288	0	512
extent i/o buffers	0	2147483647	0	0
identity burning set factor	1	9999999	5000	5000
allow sendmsg	0	1	0	0
sendmsg starting port number	0	65535	0	0

Physical Memory Utilization Scheme

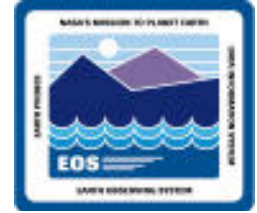




Topology for Running Two Event Processors



Sybase Security (Auditing)



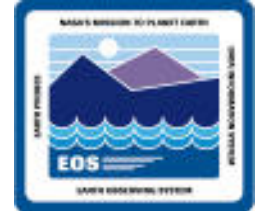
- 1) Run sybinit and install auditing.
- 2) Add a login for auditing:

```
sp_addlogin ssa, ssa_password, sybsecurity
use sybsecurity
sp_changedbowner ssa
sp_role "grant", sso_role, ssa
```
- 3) Enable auditing:

```
sp_auditoption "enable auditing", "on"
sp_auditlogin loginname, "cmdtext", "on"
```
- 4) To Test:

```
create a table in a database with one field
grant all on the table for the loginname
log into isql using the loginname
insert a record into the table
log into isql as ssa
select * from sysaudits where loginname = "loginname"
```

Integrity Monitoring



Database Consistency Checker

dbcc: is a set of utility commands for checking the logical and physical consistency of a database